

Vom Einsteiger zum Profi

Programmieren

- ★ Floppy und Datasette
- ★ Großer Kurs: Basic von A bis Z

Viele Tips & Tricks für den C 64

Super Listings

- ★ Tolles Action-Spiel
- ★ Schnelle Hires-Grafik
- ★ Professionelle Anwendungen

Alle Programme auch auf Diskette erhältlich



Jetzt geht's aufwärts



Es ist noch kein Meister vom Himmel gefallen! Auch nicht bei den Computer-Freaks. Ganz im Gegenteil, gerade die ersten Schritte in die Computertechnik fallen einem nicht leicht. Wir wollen Ihnen mit diesem 64'er-Sonderheft nicht nur helfen, die ersten Hürden zu überwinden, sondern der Einstieg in die C64-Welt soll so leicht wie möglich sein und natürlich Spaß machen.

Ein ausführlicher Basic-Kurs wird Sie bei den »ersten Schritten« mit Ihrem neuen Computer begleiten und auf interessante Weise auch all die lästigen kleinen Fehler aufdecken, die anfangs häufig gemacht werden, denn Programmieren soll ja ein Hobby werden, das Ihnen Spaß macht. »Programmieren« – sicher ist dies für Sie noch ein Zauberwort, das mit etlichen Rätseln verbunden ist. Aber anhand des Basic-Kurses werden Sie merken, daß damit auch schnelle Erfolgserlebnisse verbunden sind.

»Wissen ist Macht« heißt es so treffend. Ein Computerlexikon, das sicher noch lange ein wichtiges Nachschlagewerk bleibt, wird Ihnen immer wieder helfen, weitere Erkenntnisse über den Computer zu erlangen. Von A bis Z finden Sie Fachbegriffe verständlich erklärt.

Wenn Sie erst ein paar Programme selbst geschrieben haben, wird es Ihnen sicher bald lästig, daß ein Programm jedesmal, nachdem Sie den Computer ausgeschaltet haben, gelöscht ist. Also, ein Massenspeicher wie ein Diskettenlaufwerk oder eine Datasette muß her. Der Umgang mit Disketten bietet erheblich mehr Komfort als eine Datasette, zumal bei Commodore-Laufwerken eine »intelligente« Elektronik integriert ist. Die Steuerelektronik des Diskettenlaufwerks kann eine Vielzahl von Fehlern selbst analysieren und dem Computer mitteilen – unabhängig, ob die Diskette fehlerhaft ist oder Sie das Gerät falsch bedient haben. Auf dem Monitor erscheint dann eine mit Code-Zahlen vermischte und zumeist ohnehin unverständliche Fehlermeldung. Wir zeigen Ihnen, was all diese Meldungen zu bedeuten haben und wie Sie die meisten Fehler verhindern können. Wenn Sie Zugriffe auf die Diskette in eigene Programme selbst einbinden wollen, werden Sie sehr bald merken, daß Ihr Programm bei jeder Fehlermeldung »aussteigt«. Wie so etwas geschickt zu verhindern ist, erfahren Sie ebenfalls.

Zum Thema »Programmieren« gibt es natürlich eine Menge Tips & Tricks. Besonders über PEEKs und POKEs lassen sich diverse Probleme einfach lösen, die in der normalen Basic-Programmierung nur sehr schwer oder gar nicht bewältigt werden können. Und warum sollen Sie »das Rad noch einmal erfinden«, wenn wir dank der Unterstützung unserer 64'er-Leser auf einen mehrjährigen Erfahrungsschatz zurückgreifen können.

Für Sie haben wir diese PEEKs und POKEs zusammengefaßt und übersichtlich geordnet, damit Sie den entscheidenden Trick schnell finden können.

Natürlich wollen Sie sich am Computer ab und zu einfach entspannen. Was ist dazu besser geeignet als ein Computerspiel? Zwei tolle Spiele haben wir ausgesucht, die Sie nur abzutippen brauchen. Dank unserer Eingabehilfen lassen sich fast alle Fehler beim Abtippen vermeiden. Wie Sie mit den Eingabehilfen umgehen müssen, finden Sie auf den letzten Seiten ausführlich erklärt.

Sicher war für Sie auch ein Grund, sich für den C64 zu entscheiden, daß es für diesen Computer schon eine enorme Anzahl von ausgereifter Software gibt, und Sie bei Freunden oder Bekannten schon einige dieser faszinierenden Programme kennengelert haben. Man will ja seinen Computer auch sinnvoll nutzen. Aus dem inzwischen kaum zu überschauenden Software-Angebot haben wir für die wichtigsten Anwendungsbereiche jeweils ein Spitzenprogramm herausgesucht, das Ihnen gute Dienste leisten kann.

Dank des Erfindungsreichtums der Leser konnten wir im 64'er-Magazin etliche Programme zum Abtippen anbieten, die den Vergleich zu kommerzieller Software nicht zu scheuen brauchen und diese sogar übertreffen. In einer Übersicht finden Sie eine Reihe unserer Programm-Renner beschrieben, die insgesamt eine hervorragende Grundausstattung ergeben. Computergrafiken waren dabei schon immer etwas Fesselndes. Eine der schnellsten Grafikerweiterungen, die es für den C64 gibt, können Sie sogar abtippen.

Wie Sie sicher schon gemerkt haben, wollen wir Ihnen mit diesem Sonderheft für den Einstieg in die »Computerei« das wichtigste Handwerkszeug zur Verfügung stellen. Natürlich kann ein Sonderheft nicht alles enthalten, was man gebrauchen könnte. Deshalb erscheinen zu den meisten Themen mehrere Sonderhefte, von denen jedes in sich selbst abgeschlossen ist. Ich möchte Sie daher noch auf das Sonderheft 16 aufmerksam machen, das ebenfalls für C64-Einsteiger gedacht ist und diese Ausgabe hervorragend ergänzt.

Ihr Gottfried Knechtel (Redakteur)


PROGRAMM-
SERVICE

INHALT

Einleitung

Software

Spiele

Grundlagen

Grafik-Listings

Tips & Tricks

Eingabehilfen

Sonstiges

Alle Programme aus Artikeln mit dem ■ -Symbol finden Sie auf der Programmservice-Diskette zu diesem Sonderheft (siehe linke Seite)

Nur vom Besten...



Für den C64 gibt es eine kaum noch überschaubare Anzahl an Programmen für jeden erdenklichen Zweck. Besonders für den Einsteiger wird es daher immer schwerer, das für ihn beste Programm herauszufinden. Grund genug, die stärksten professionellen Programme für den C64 kurz vorzustellen.

Im 64'er-Magazin wurden seit der Erstausgabe im April 1984 bereits sehr viele Programme ausführlich getestet. Doch wie muß ein Programm aussehen, um zur Referenz-Software gekürt zu werden? Der wohl mit Abstand wichtigste Aspekt ist natürlich die Qualität des Produktes und sein Preis-/Leistungs-Verhältnis. In einigen Bereichen stellen wir Ihnen zwei Programme vor, die sich entweder im Preis deutlich voneinander unterscheiden oder aber in bezug auf den geeigneten Einsatzbereich deutliche Unterschiede aufweisen.

Produkt/Anbieter	Test in Ausgabe	Preis:
Vizawrite 64	10/84	98 Mark
Vizastar 64 DTM Werbung und EDV Bornhofenweg 5, 6200 Wiesbaden	11/84	298 Mark als Paket mit Vizawrite 64 348 Mark
Startexter 64	9/85	64 Mark
Stardatei 64	4/86	64 Mark
Starpainter 64 Sybex Verlag, Vogelsanger Weg 111, 4000 Düsseldorf 30	10/86	64 Mark
Hi-Eddi Plus	5/86	48 Mark
Giga CAD Plus Markt&Technik Verlag AG Hans-Pinsel-Str. 2 8013 Haar bei München	2/87	49 Mark
Superbase 64	5/84	99 Mark
Basic 64 Data Becker, Merowinger Str. 30 4000 Düsseldorf 1	4/85 und SH 12	99 Mark
VIP-Terminal Klaus F. Erbrecht, Lappenbergsallee 37, 2000 Hamburg 20	Marktübersicht 4/87	51,30 Mark
Sound-Expander Side by Side Landgraf-Philipp-Str. 65 6000 Frankfurt 50	9/87	399 Mark Composer/Editor 120 Mark Keyboard 280 Mark
Printfox Scantronic, Parkstr. 38, 8011 Zorneding	6/86	98 Mark
Print-Shop Softline, Schwarzwaldstr. 8a, 7602 Oberkirch	4/85	129 Mark
ASSI/M Dirk Zabel, Stresemannstr. 50, 1000 Berlin 61	1/85	ab 220 Mark

Eine Grundvoraussetzung für die Software-Referenz ist aber auch, daß das Programm in Deutschland vertrieben wird und auch erhältlich ist. In vielen Fällen haben jedoch auch unsere Leser entschieden. Durch unsere Umfrage-Wettbewerbe wissen wir, welche Programme besonders stark verbreitet sind und besonders häufig eingesetzt werden, also am beliebtesten sind.

Auf den nachfolgenden Seiten finden Sie für die gängigen Einsatzgebiete des C64 besonders empfehlenswerte Programme. Aus Platzgründen ist es uns an dieser Stelle natürlich nicht möglich, alle Leistungsmerkmale und integrierten Funktionen aufzuzeigen.

In vielen Fällen bieten jedoch die Hersteller oder Verteiler der Programme Informationsblätter an. Sicherlich ist nicht jedes Programm für jeden Anwender uneingeschränkt geeignet. Viele Programme fordern eine bestimmte Hardware-Grundausstattung, zum Beispiel ein Diskettenlaufwerk, einen Joystick oder bestimmte Druckertypen. Informieren Sie sich daher bitte vor der Kaufentscheidung, ob das für Sie interessante Programm mit Ihrer Hardware-Konfiguration ohne Probleme läuft. Dazu entnehmen Sie bitte aus der Tabelle die Ausgabe der 64'er, in der diese Programme getestet wurden, sowie die Anschrift der Anbieter dieser Produkte, über die Sie nähere Informationen beziehen können.

Einige der hier aufgeführten Programme werden wir in der Ausgabe 8/87 nochmal ausführlicher vorstellen. So können Sie sich ein klares Bild darüber machen, was die Programme im praktischen Einsatz leisten.

An anderer Stelle in diesem Heft finden Sie auch eine Aufstellung der 64'er Top-Listings zum Abtippen. In vielen Fällen können Sie mit diesen Programmen vergleichbare Ergebnisse erzielen. Beachten Sie also auch diese interessante Zusammenstellung.



Vizawrite 64

Textverarbeitung

»Vizawrite« 64 zeichnet sich vor allem durch eine hohe Geschwindigkeit, einen großen Textspeicher (34 KByte) und komfortable Editier- und Formatierungsfunktionen aus. Da mit einem Trick eigene Programmerweiterungen nachgeladen werden können, ist Vizawrite 64 in vielen Punkten eigenen Bedürfnissen anpaßbar. Eine bisher 14teilige Serie »Tips&Tricks zu Vizawrite 64« finden Sie in loser Folge seit Ausgabe 12/85. Zum Beispiel einen nachladbaren Taschenrechner (4/87) und eine Erweiterung, die es erlaubt, Text und Grafik in Briefqualität auf einem Matrixdrucker auszugeben (Listing des Monats 6/87).



Startexter 64

Textverarbeitung II

Nicht nur wegen des günstigen Preises gehört »Startexter 64« zu den verbreitetsten Programmen bei unseren Lesern. Startexter erlaubt es ohne weiteres, auf problematischen Druckern Umlaute auszugeben. Einer der größten Vorteile von Startexter 64 ist der 80-Zeichen-Modus. Hier kann der Text in der Form betrachtet werden, wie er später auf dem Papier ausgegeben wird. Startexter verarbeitet auch Trennvorschläge, die dafür sorgen, daß beim Blocksatz keine großen Lücken zwischen den Wörtern einer Zeile entstehen. Mit dem mitgelieferten Editor können Bildschirm- und Drucker-Zeichensätze entworfen werden.



Stardatei 64

Dateiverwaltung

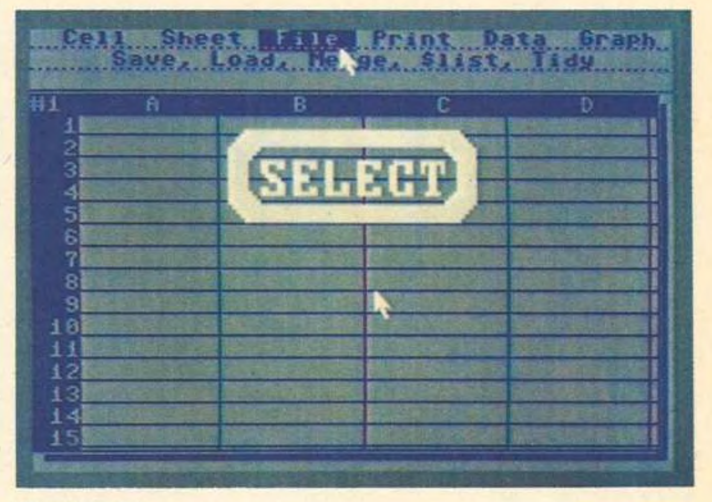
Für viele Anwendungen im Bereich Dateiverwaltung ist es ausreichend, mit »elektronischen Karteikarten« zu arbeiten. »Stardatei 64« ist ausgesprochen einfach zu bedienen, da es letztlich in der Handhabung einer konventionellen Kartei entspricht. Es gibt hier ebenfalls Karteikarten-Reiter, nach denen besonders schnell gesucht werden kann. Der Inhalt einer bestimmten Karteikarte kann so schnell und einfach eingesehen und ausgegeben werden. Interessant ist sicherlich auch die Schnittstelle zu Startexter 64, die es erlaubt, gefundene Datensätze in das Textverarbeitungsprogramm zu übertragen.



Superbase 64

Datenbank-System

Das wohl einzige Programm für den C64, das zu Recht den Namen Datenbank trägt, ist »Superbase 64«. Neben einer komfortablen menügesteuerten Datenverarbeitung ist es mit diesem Programm auch möglich, Datensätze mit Hilfe einer eigenen Datenbank-Programmiersprache zu manipulieren. So ist Superbase 64 gleichermaßen für den Anfänger und den Profi geeignet. Da auch komplexe Berechnungen durchgeführt werden können, ist Superbase 64 für nahezu jedes Problem einsetzbar – von der einfachen Adressenverwaltung bis hin zur Lager- und Finanzbuchhaltung.



Vizastar 64

Tabellenkalkulation

Im professionellen Einsatz des C64 spielt auch die Tabellenkalkulation eine wichtige Rolle. »Vizastar 64« stellt alle wichtigen mathematischen Funktionen für ein professionelles Arbeiten zur Verfügung. Mit der eigenen Programmiersprache können Eingaben und Berechnungen sowohl menü- als auch programmgesteuert durchgeführt werden. Integriert ist auch eine Datenbank, mit der Daten jeder Art verwaltet und in die Tabellenkalkulation übertragen werden können. Ergebnisse können auch grafisch dargestellt und auf vielen Druckern ausgegeben werden. Eine Schnittstelle zu Vizawrite 64 ist vorhanden.



Hi-Eddi Plus



Printfox

Grafik/Konstruktion

»Hi-Eddi Plus«, die erweiterte und verbesserte Version des 64'er-Programms, arbeitet mit bis zu sieben Bildschirmen in der höchstmöglichen Auflösung des C64 (320 x 200 Bildpunkte) bei maximal vier Farben. Über den integrierten Sprite-Editor kann jeder Bildpunkt exakt angesprochen werden. Bilder vieler Zeichen- und Malprogramme sowie Zeichensätze und Sprites können geladen und weiterverarbeitet werden. Alle wichtigen Zeichenbefehle und eine schnelle Trickfilm-Routine sind vorhanden. Darüber hinaus können mit Hi-Eddi Plus auch Spraydosen-Effekte erzielt und Beschriftungen vorgenommen werden.

Desk-Top-Publishing

Wer mehrspaltige Texte in verschiedenen Schriftarten und -größen mit Grafik kombiniert verarbeiten möchte, ist mit dem »Printfox« gut beraten. Dieses Programm enthält ein leistungsfähiges Textverarbeitungsprogramm, das Vizawrite 64 nachempfunden ist, und einen Grafikeditor, der stark an Hi-Eddi erinnert. Durch diese gelungene Kombination ist Printfox zum Gestalten von Schüler- und Vereinszeitungen oder auch für kürzere Briefe optimal geeignet. Zusatzprogramme zum Entwerfen neuer Zeichensätze sowie weitere Grafikdisketten zum Einbinden in die Texte machen das Programm noch interessanter.



Starpainter 64



Printshop

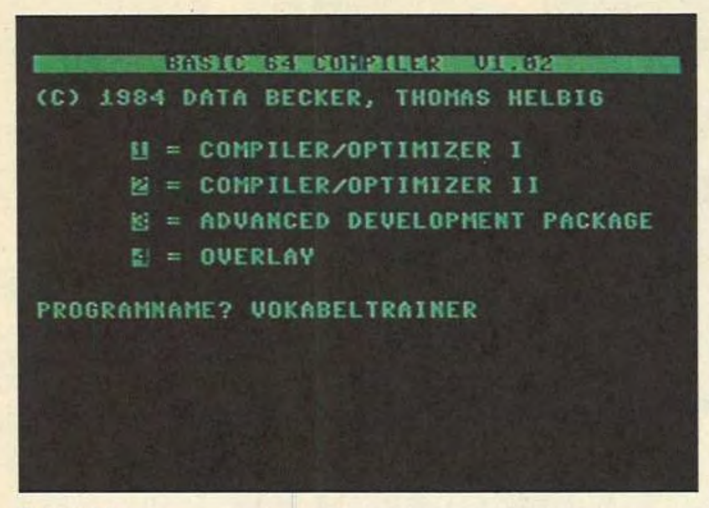
Grafik/Konstruktion II

Im Gegensatz zu vielen anderen Zeichen- und Konstruktionsprogrammen arbeitet »Starpainter 64« im DIN-A4-Format. Das heißt, daß letztlich eine Grafik auf einem Drucker ausgegeben werden kann, die die gesamte Fläche einer Seite ausnutzt. Während des Zeichnens sieht man selbstverständlich nur einen Teil des gesamten Bildes, allerdings kann man sich jederzeit das Gesamtbild in stark verkleinertem Maßstab anzeigen lassen. Viele verschiedene »Pinsel« und Füllmuster stehen für die Arbeit mit Starpainter zur Verfügung. Für kompliziertere Konstruktionen kann ein »Lineal« als Maßstab eingeblendet werden.

Drucker-Software

Auf den schnellen Entwurf und Druck von Gruß- und Glückwunschkarten, Schildern (DIN A4), Briefköpfen, Kalenderblättern und meterlangen Transparenten ist der »Print-Shop« spezialisiert. Für den Text stehen acht verschiedene Proportionalschrift-Zeichensätze zur Verfügung, die gefüllt, umrandet und im 3D-Effekt verwendet werden können. Die Bilder können mit einem von acht verschiedenen Rahmenmuster umrandet werden und mit einem der 60 mitgelieferten oder auch selbstentworfenen kleinen Bildern versehen werden. Als Hintergund können auch beliebige Grafikbilder eingebunden werden.

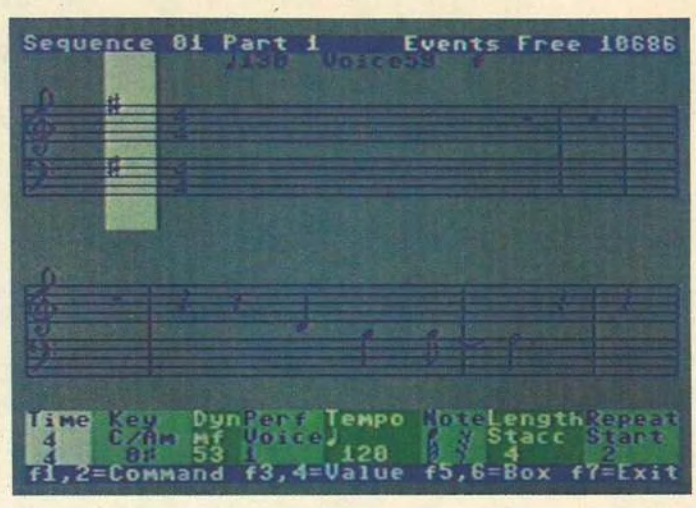




Basic 64

Basic-Compiler

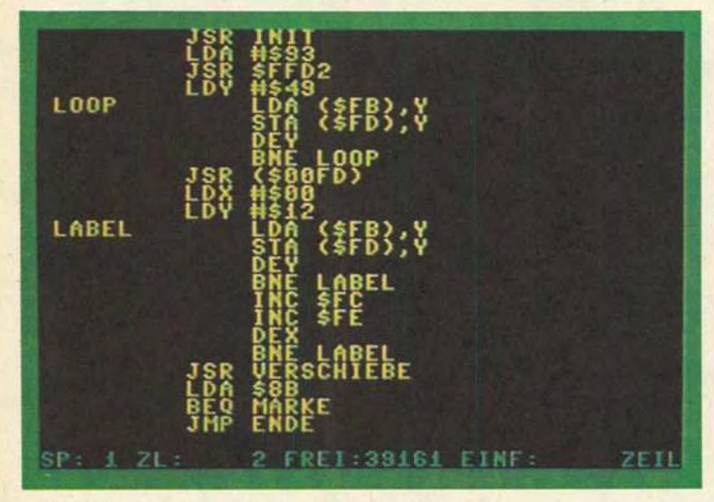
Wer viel in Basic arbeitet, aber nicht auf die Geschwindigkeit maschinennaher Programmierung verzichten möchte, benötigt einen guten Compiler. Der schnellste und vielseitigste Compiler für den C64 ist »Basic 64«. Das interessante an Basic 64 ist die Funktion, bei der Basic-Befehle in echte Maschinensprache übersetzt werden (M-Code). Wahlweise kann auch der platzsparende P-Code eingestellt werden, der jedoch etwas langsamer abgearbeitet wird. Auch das Mischen der beiden Codes ist möglich. Das Compilat kann frei im Speicher verschoben werden, was in vielen Fällen nützlich ist.



Sound Expander

Musik

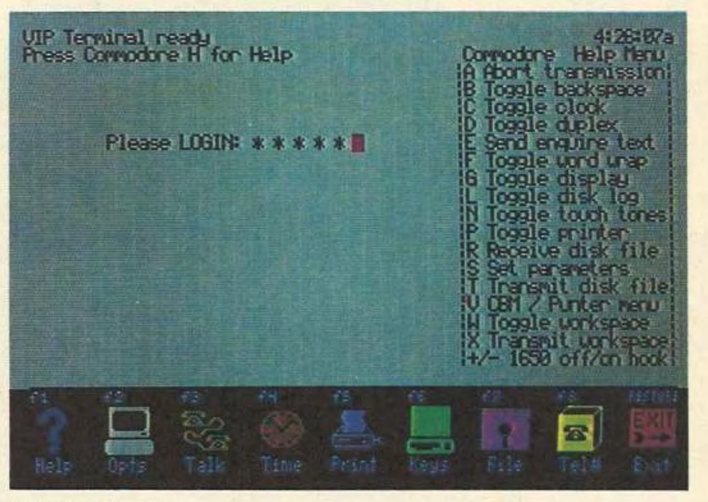
Wer mehr als die drei Stimmen des C64 zur Musikprogrammierung benötigt, kann mit dem »Sound-Expander« bis zu neunstimmig polyphon spielen. Spezielle Software zur Musik- und Sound-Editierung (Bild) und -Programmierung sowie ein professionelles Keyboard können zusätzlich einzeln erworben werden. Herz des Sound-Expanders, der in den Expansions-Port des C64 gesteckt wird, ist ein Yamaha-Sound-Chip, der den C64 zum professionellen Synthesizer macht. Das Besondere am eingebauten Sound-Chip ist der besonders klare Klang (FM-Sound) mit dem Unglaubliches aus dem C64 herausgeholt werden kann.



ASSI/M

Assembler

Für die Programmierung in Maschinensprache ist ein Assembler notwendig. »ASSI/M« ist ein Paket, bestehend aus Full-Screen-Editor (FSE), Monitor (DEMON) und Assembler (ASM). Der Editor benötigt keine Zeilennumerierung und erlaubt Blockstrukturierungen durch Einrückung. Ähnlich wie bei einem Textverarbeitungsprogramm gibt es leistungsfähige Befehle zur Bearbeitung. Der Assembler ist besonders schnell und verarbeitet Makros, so daß Makrobibliotheken angelegt werden können (zwei werden mitgeliefert). Besonders leistungsfähig und vielseitig ist auch der Maschinensprache-Monitor.



VIP-Terminal

DFÜ/Terminalprogramme

Wer über einen Akkustikkoppler oder Modem verfügt, kann über »VIP-Term« mit anderen Computern, Mailboxen und Datenbanken in einer Darstellung von 40, 80 oder 106 Zeichen/Zeile kommunizieren. Als einziges bekanntes Programm bietet VIP-Term für den C64 lauffähige VT-52-Terminal-Emulation, die Sie in vielen professionellen Mailboxen benötigen, sowie eine Übertragung mit 1200 Baud, 8 Datenbits, keiner Parität und 1 Stoppbit (1200/8/n/1). VIP-Term besitzt weiterhin als nützliche Funktion ein X-Modem-Übertragungsprotokoll, so daß auch Programmdateien per Telefon übertragen werden können.

Die »Stars« der 64'er

Viele Programme, die im 64'er-Magazin oder den Sonderheften veröffentlicht wurden, entwickelten sich zu wahren Standardprogrammen für den C64. Wir stellen Ihnen eine kleine Auswahl unserer »Hits« vor, die in keiner Software-Bibliothek fehlen sollten.

Viele Spitzenprogramme wurden im 64'er-Magazin schon als »Listing des Monats« oder »Anwendung des Monats« veröffentlicht. Doch auch unter den weniger spektakulären Programmen waren viele aus den unterschiedlichsten Richtungen, die so manchem professionellen und teuren Produkt mühelos das Wasser reichen können oder es gar übertreffen.

Wir wollen nun in diesem Bericht die »Sternstunden« des 64'er-Magazins noch einmal Revue passieren lassen, um Ihnen zu zeigen, zu welchen Leistungen unsere Listings imstande sind. Wenden wir uns zunächst den Programmen zu, die dem Umgang mit dem Floppylaufwerk dienen:

Die Floppy 1541 ist als Speichermedium den meisten C64-Anwendern bekannt und wird neben der Datasette am häufigsten zur Speicherung von Programmen und anderen Daten verwendet. Doch im Vergleich zu den Diskettenstationen anderer Computersysteme ist das Floppylaufwerk des C64 sehr langsam. Dies macht sich besonders beim Laden längerer Programme bemerkbar. Aus diesem Grund entstand »Hypra-Load« – ein kleines Programm von etwa 1500 Byte Länge, das den Ladevorgang der Floppy 1541 etwa um den Faktor 5 beschleunigt. Als erster Floppy-Speeder seiner Art wurde »Hypra-Load« auch sofort als Listing des Monats in der Ausgabe 10/84 des 64'er-Magazins vorgestellt. »Hypra-Load« war eine kleine Sensation und der Auslöser für viele weitere Hypra-Projekte.

Kurze Zeit später folgte in Ausgabe 4/85 ein Listing, mit dem man die schnelle Lade-Routine per Austausch eines EPROMs in das Betriebssystem des C64 einbauen konnte, denn Hypra-Load mußte vor seinem Einsatz stets neu geladen werden. Befinden sich die nötigen Programmteile schon im Betriebssystem des Computers, steht es bereits nach dem Einschalten des C64 zur Verfügung und kann sofort seine vollen Geschwindigkeitsvorteile entfalten. Das »neue« Betriebssystem kam in Ausgabe 4/85 zu Veröffentlichung. Es bekam zu Ehren seines Vorgängers den Namen »Hypra-Perfekt«.

Die »Hypra-Welle«

Doch warum sollte man Programme nicht auch 5mal schneller speichern können? Als Ergänzung zu Hypra-Load wurde deshalb »Hypra-Save« veröffentlicht, mit dessen Hilfe der Speichervorgang um den Faktor 5 beschleunigt wird. Es wurde in der Ausgabe 8/85 abgedruckt.

Neueren Datums ist dagegen das dritte Programm im »Hypra«-Bunde mit dem Namen »Hypra-Format«. Es wurde erst kürzlich in Sonderheft 15 des 64'er-Magazins unter der Rubrik »Tips & Tricks« veröffentlicht. Jede neue Diskette muß formatiert werden, bevor man Daten darauf speichern kann. Im Originalzustand benötigt die Floppy 1541 jedoch etwa 90 Sekunden für diese wichtige Prozedur. Doch Hypra-Format nimmt diese Arbeit in nur 15 (fünfzehn) Sekunden vor. Hypra-Format unterdrückt auch das Anschlagen des Schreib-Lese-Kopfes, das man beim Beginn des Formatierens als lautes Rattern hören kann. Außerdem übertrifft Hypra-Format beim Formatieren selbst die teuersten Floppy-Speeder in der Geschwindigkeit!

Alle Programme der Hypra-Serie funktionieren auch mit der neuen Floppy 1541c, es treten keine Kompatibilitätsschwierigkeiten auf.

EXOS V3 – Es geht noch schneller

Mit Hypra-Load, dachte man, war die Grenze des technisch Machbaren erreicht, wenn man ohne Hardware-Zusätze auskommen will. Doch Ende 1986 wurde man eines Besseren belehrt. Ein neues Betriebssystem für den C64 mit dem Namen »EXOS V3«, das, auf EPROM gebrannt, gegen das alte ausgetauscht wird, enthält einen Schnellader, der es erlaubt, Programme bis zu 14mal schneller zu laden. Das Listing des Monats für die Ausgabe 12/86 stand somit fest.

Neben den schnellen Laderoutinen bietet EXOS V3 jedoch weitere Besonderheiten. Zusätzlich unterhält EXOS V3 eine RAM-Floppy und verfügt über viele weitere Befehle und Kommandos, die die Arbeit mit dem Bildschirmeditor des C64 erleichtern sollen. Hinzu kommen die Möglichkeiten zur Anzeige des Disketten-Directorys und des Floppy-Fehlerkanals.

Sollte das Laden mit EXOS V3 einmal Probleme bereiten, kann das Betriebssystem stufenweise bis zum ursprünglichen Original des C64 zurückgeschaltet werden, so daß keine Kompatibilitätsprobleme auftreten können.

Rekord beim Kopieren

Häufig steht man vor dem Problem, eine ganze Diskette duplizieren zu müssen. Doch oft mangelt es an einem entsprechenden Programm oder, wenn vorhanden, arbeitet es nervenaufreibend langsam. »Master-Copy«, das Listing des Monats in der Ausgabe 5/87 des 64'er-Magazins, stellt bei den reinen Software-Lösungen einen neuen Rekord in bezug auf Kopiergeschwindigkeit auf. Es kopiert eine vollständige Diskette in nur 1½ Minuten. Anders als bei »Copy+«, dem Listing des Monats in Ausgabe 3/87, legte der Autor von Master-Copy Wert auf das schnelle Kopieren ungeschützter oder einwandfreier Disketten. Fehler auf einzelnen Spuren oder Sektoren werden zwar erkannt und dem Anwender mitgeteilt, jedoch nicht auf die Zieldiskette übertragen. Selbstverständlich verfügt Master-Copy auch über ein bedienerfreundliches Menü. Darüber lassen sich Floppy-Kommandos senden und das Directory anzeigen. Weitere Menüpunkte erlauben das genaue Einstellen der Start- und Endspur und der Geräteadresse der betreffenden Floppystation. Somit können auch Diskettenlaufwerke der Nummern 9, 10 oder 11 problemlos angesprochen werden.

Master-Copy kopiert je nach angegebenem Bereich von nur einer bis zu 40 Spuren einer Diskette, und das in einer sagenhaften Zeit.

Master-Copy ist in gewisser Weise eine gute Ergänzung zu Copy+. Sollen Disketten schnell und einfach dupliziert werden, verwendet man am besten Master-Copy. Manipulierte oder fehlerhafte Disketten sind dagegen mit Copy+ zu kopieren. So hat der Anwender mit beiden Programmen für alle Fälle ein gutes Kopierprogramm zur Hand. Ein





Bild 1. Das Hauptmenü des »Disc-Wizard«

Nachteil von Copy+ soll nicht verschwiegen werden: Es arbeitet nicht mit der Floppy 1541c zusammen.

Weniger Geschwindigkeit und gutes Timing, als Komfort im Diskettenbetrieb lag dem Autor des »Disc-Wizard« am Herzen. Sein Programm ist in bezug auf Diskettenbehandlung wahrlich ein Zauberer. Aus diesem Grund wurde es auch in der Ausgabe 5/86 des 64'er-Magazins das Listing des Monats. Es ist unter anderem auch in Sonderheft 15 (Thema: »Floppy und Datasette«) zu finden.

Der Zauberer für die Floppy 1541

Mit dem Disc-Wizard hält man kein gewöhnliches Utility-Programm in den Händen, sondern hat die Wahl zwischen vielen ungewöhnlichen, aber wertvollen Möglichkeiten zur Manipulation von Disketten. So lassen sich auf einfache Weise der Name und die ID einer Diskette auch ohne eine Neuformatierung ändern oder ganze Disketten gegen Schreibzugriffe schützen. Sollen nur einzelne Dateien vor dem Überschreiben geschützt werden, kann auch dies mit Disc-Wizard problemlos und einfach über ein komfortables Menü (Bild 1) vorgenommen werden.

Haben Sie eine Diskette irrtümlich kurz formatiert (Floppybefehl N ohne ID-Angabe), obwohl Sie die darauf gespeicherten Programme noch benötigt hätten? In einem solchen, bisher aussichtslosen Fall hilft Disc-Wizard mit einem hervorragenden Befehl, die angeblich gelöschten Dateien wieder zu rekonstruieren. Dies ist jedoch keine Zauberei, denn bei der Kurz-Formatierung wird lediglich das Inhaltsverzeichnis einer Diskette gelöscht; die restlichen Daten bleiben erhalten. Bei mit Angabe der ID formatierten Disketten hat diese Methode keinen Erfolg, da dabei sämtliche Daten vollständig überschrieben werden.

Besonders im Zusammenhang mit dem Directory leistet der Disc-Wizard Ungewöhnliches. Nahezu alle dort verzeichneten Parameter lassen sich beliebig ändern, manipulieren und verdrehen, wie beispielsweise der Name, der Typ oder die Längenangabe einer Datei. Mit der Anweisung »DIR-SORTER« können sogar Dateien im Directory sortiert, eingefügt oder gelöscht werden.

Des weiteren ist im Disc-Wizard ein umfangreicher Disketten-Monitor implementiert, der so manches Produkt in den Schatten stellt. Hier können die einzelnen Sektoren einer Diskette nach Belieben geändert werden. Eine besondere Fähigkeit dieses Programms besteht darin, die Daten eines Blocks auf der Diskette nach, wenn es sein muß auch verschlüsselten, Texten zu durchsuchen, um eventuell codierte Daten aufzuspüren. Natürlich können die Daten nach Herzenslust selbst codiert werden, so daß sie für Unbefugte nicht lesbar sind.

Disc-Wizard ist ein Programm, das in der Software-Sammlung eines Floppy-Fans nicht fehlen sollte.

Wie Sie bisher sehen konnten, haben wir uns beim 64'er-Magazin über das Thema Disketten und Floppystation sehr viele Gedanken gemacht. Doch auch in anderen Bereichen wurden sehr gute Listings veröffentlicht.

Grafisch Up-to-Date

Gleich zu Beginn des Jahres 1985, in der Ausgabe 1/85, sorgte ein fantastisches Listing des Monats für Furore. Sein Name ist »Hi-Eddi«, der als Abkürzung für »High Resolution-Graphics-Editor« steht. Es ist ein professionelles Zeichen- und Malprogramm für den C64, das sich hinter kommerzieller Software nicht verstecken muß. Hans Haberl, der Autor von Hi-Eddi wollte ursprünglich nur chinesische Zeichen auf dem C64 darstellen. Dann erweiterte er seinen Editor aber schließlich zu einem perfekten Grafikprogramm mit Menüsteuerung (Bild 2), das mit einer Vielzahl von Kommandos Möglichkeiten eröffnet, die man bei so manchem Grafikeditor vermißt. Befehle zum Zeichnen von Linien, Rechtecken und Kreisen sind für Hi-Eddi genauso selbstverständlich wie das Ausfüllen beliebiger Flächen mit einer gewünschten Farbe oder das Bewegen von einzelnen Bildausschnitten. Einige Besonderheiten zeigen aber, daß es sich hier nicht um ein gewöhnliches Zeichenprogramm handelt.

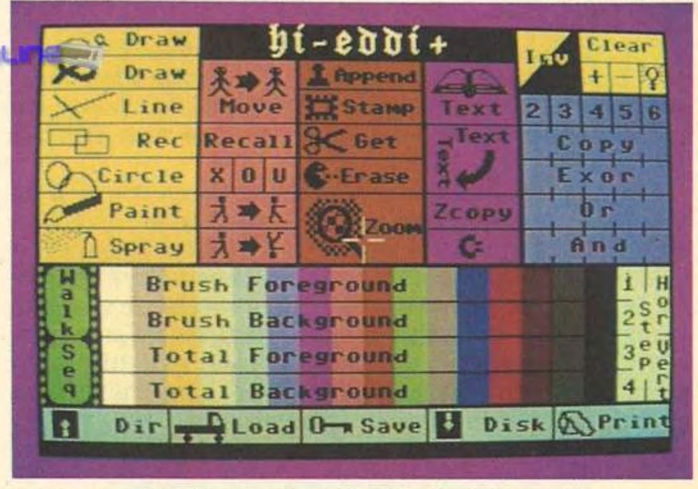


Bild 2. Das Menü von »Hi-Eddi«. Alle Funktionen können per Joystick angewählt werden

So besitzt Hi-Eddi bis zu sieben unabhängige Grafikbildschirme (Grafikseiten), zwischen denen wahlweise umgeschaltet werden kann. Der Computergrafiker verfügt damit über eine außerordentlich große Arbeitsfläche zur Erstellung seiner Werke. Natürlich können einzelne Grafikabschnitte zwischen diesen Bildschirmen beliebig kopiert werden. Selbst ganze Grafikseiten dürfen in ein anderes Bild übertragen werden. Hi-Eddi kann dabei auf Wunsch eine Grafik wie eine Folie auf eine andere legen, so daß diese nach bestimmten Vorschriften vereinigt und kombiniert werden. Der Anwender hat somit die Möglichkeit, Teile seines Bildes - zum Beispiel Hintergrund und Vordergrund - getrennt zu entwerfen, um sie schließlich zu einem Gesamtbild zusammenzufügen.

Doch Hi-Eddi ist nicht nur ein komfortables Zeichenprogramm. Es ist vielmehr auch ein perfekter Sprite-Editor, der

das Gestalten von Sprites – dies sind frei bewegliche Grafikobjekte, wie man sie von Spielen her kennt – zum Kinderspiel macht. Im Vergrößerungsmodus kann ein Sprite Punkt für Punkt eingegeben werden. Ein Sprite läßt sich aber auch jederzeit aus einem Grafikbild herauskopieren, um es dann weiter zu verarbeiten. Drehen, Spiegeln und Wenden eines Sprites sind dabei für Hi-Eddi keine Schwierigkeit. Nach der Bearbeitung kann es schließlich wieder in das Grafikbild eingesetzt oder auf Diskette gespeichert werden. Auf diese Weise können sogenannte Construction-Sets erstellt werden. Ein solches Set besteht aus verschiedensten Bildern eines Themas, wie zum Beispiel den Schaltsymbolen integrierter Bausteine. Durch Herausgreifen der einzelnen Teile mit der Spritefunktion lassen sich dann die gewünschten Bilder konstruieren.

Animierte Grafik mit Hi-Eddi

Eine besondere Fähigkeit von Hi-Eddi ist eine Zeichentrick-Funktion, die es erlaubt, eine kleine Zeichentricksequenz zu erstellen und diese ablaufen zu lassen. Dazu werden bis zu sechs Bildschirme in jeweils vier Teilbereiche aufgespalten, in denen sich je eine Phase der Bewegung befindet. Ein Tastendruck genügt, und Hi-Eddi »blättert« die Einzelbilder wie einen Film durch. Bis zu 24 Bilder können einen solchen Bewegungsablauf darstellen.

Alle diese Funktionen lassen sich über die Tastatur erreichen. Wer allerdings gerne mit einem Menü arbeiten will, mit dem er die einzelnen Punkte per Joystick aktivieren kann, der darf sich eines nach persönlichen Vorstellungen kreieren. Oder man greift einfach auf das schon vorhandene Menü zurück.

Der Autor von Hi-Eddi legte großen Wert auf Flexibilität und Zusammenarbeit mit anderen Zeichenprogrammen. Aus diesem Grund lassen sich auch Grafikbilder laden, die von anderen Programmen stammen, wie zum Beispiel Koala-Painter oder Paint-Magic.

Aufgrund der hervorragenden Fähigkeiten von Hi-Eddi erweiterte und vervollkommnete Hans Haberl sein Programm, das heute unter dem Namen »Hi-Eddi Plus« in Verbindung mit einem umfangreichen Buch vom Markt & Technik Software-Verlag mit großem Erfolg vertrieben wird. Ein »einfaches« Leser-Listing wurde somit zu einem professionellen Programm. Ein ähnlicher Aufstieg widerfuhr auch einem anderen sagenhaften Programm zum Thema Grafik, das zunächst als Leser-Listing begann.

Giga-CAD – einfach gigantisch

Ein Grafikprogramm zum Erstellen von dreidimensionalen Objekten (»CAD« oder »Computer Aided Design«) mit verdeckten Linien, Schattierungen und Fluchtpunktdarstellung entwerfen?

Unmöglich! Dies war die vorherrschende Meinung, auch in Kreisen von Fachleuten, bis wir schließlich ein Programm in den Händen hielten, das den vielversprechenden Namen »GIGA-CAD« trug. Die Erwartungen waren groß, ebenso die Begeisterung, als man das Programm in Aktion sah. Es war tatsächlich ein CAD-Programm für den C64 zum Konstruieren beliebiger dreidimensionaler Körper, die sich in alle erdenkliche Richtungen drehen und verändern ließen. Selbst die Darstellung mit verdeckten Linien und Schattierung fehlte nicht. Doch wurde Giga-CAD wegen seines großen Umfangs nicht Listing des Monats.

Es fand im Grafik-Sonderheft 6/86 seinen gebührenden Platz, wo Giga-CAD zusammen mit einer umfangreichen Anleitung veröffentlicht wurde.

Dank einer bedienerfreundlichen Menüsteuerung wird das Entwickeln von dreidimensoinalen Körpern mit Giga-CAD zum Vergnügen. Der gesamte Konstruktionsvorgang kann theoretisch mit dem Joystick erfolgen. Gewisse Spezialfunktionen sind jedoch nur über die Tastatur erreichbar, sowie auch die Auswahl bestimmter Optionen in diversen Untermenüs.

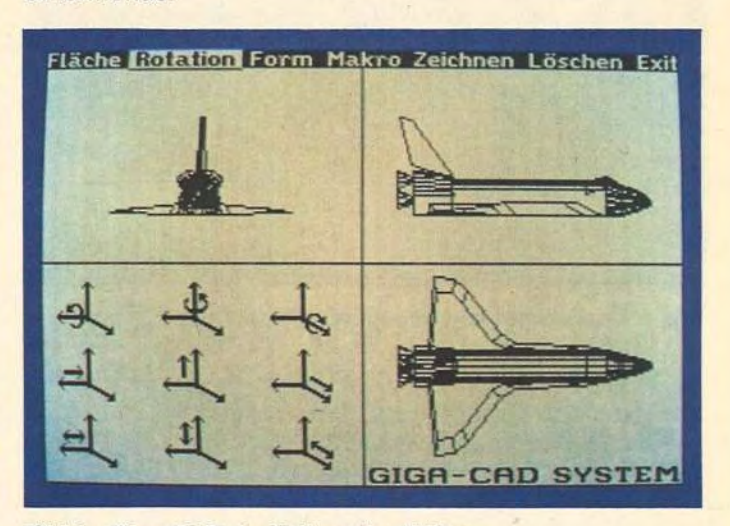


Bild 3. »Giga-CAD« in Aktion: der Editor

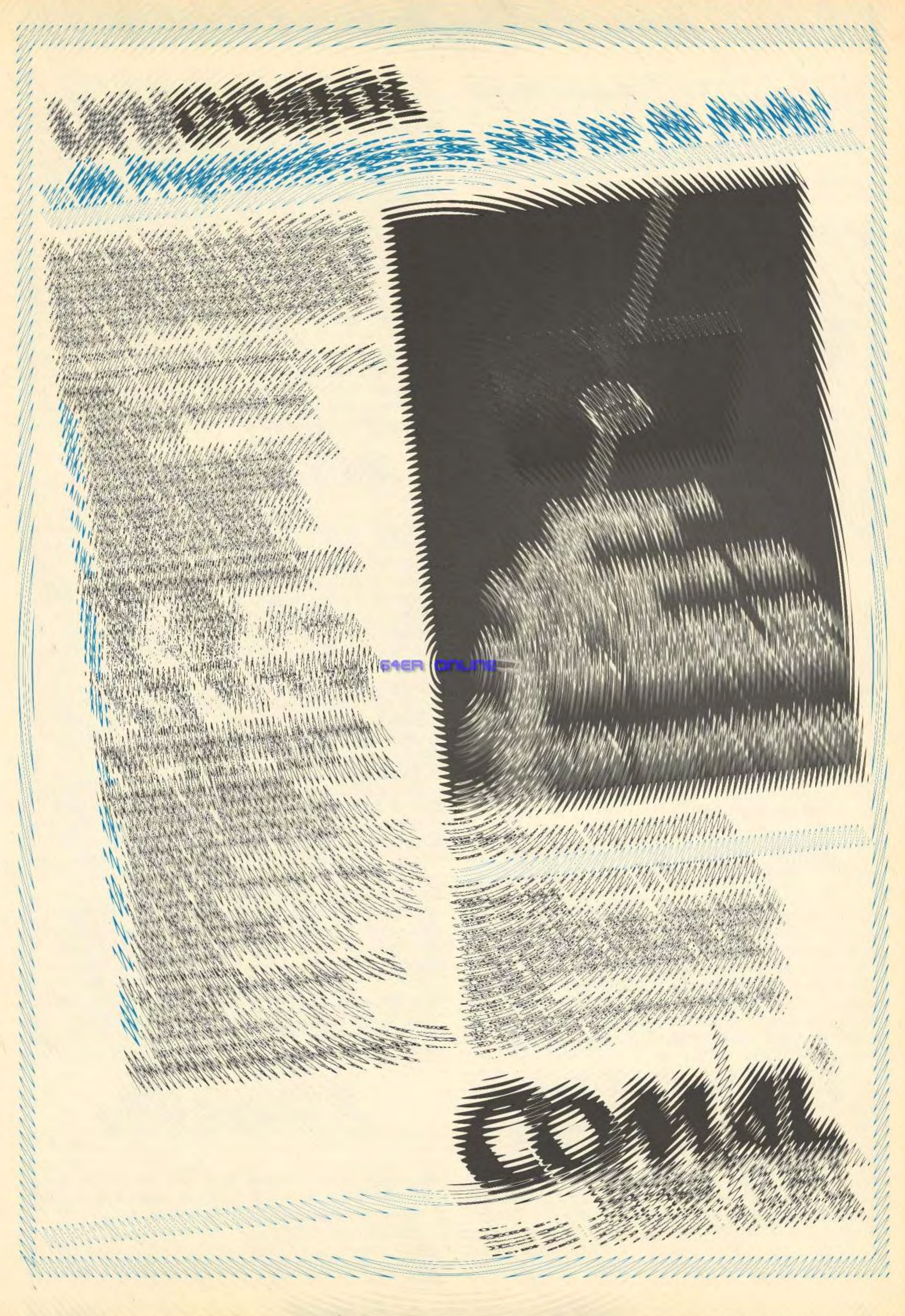
Das Erstaunliche an diesem CAD-Programm ist, daß der Körper während der Konstruktion in drei verschiedenen Ansichten (Projektionsebenen) auf dem Bildschirm sichtbar ist (Bild 3). Man kann sein Werk also stets von mehreren Seiten betrachten.

Ein Objekt bei Giga-CAD besteht aus verschiedenen Flächen, die einzeln entworfen werden. Nachdem man die Fläche in die richtige Lage gedreht hat, kann man sie schließlich an den Gesamtkörper anfügen. Damit sind beliebige unregelmäßige Objekte denkbar. Symmetrische Gegenstände lassen sich jedoch am einfachsten durch Rotation erzeugen. Hier wird ein Linienzug um eine Achse rotiert und bildet somit einen vollständigen Körper, wie Sie in Bild 4 sehen können. Daneben lassen sich auch synchron in allen drei Projektionsebenen direkt Veränderungen anbringen.

Hat man sein Objekt schließlich entworfen, kann man dessen Darstellungsweise ändern, denn noch ist es ein durchsichtiges Strichgerüst. In speziellen Menüs können die Modi zum Zeichnen mit versteckten Linien oder Schattierungen gewählt werden. Daneben ist auch die Fluchtpunktdarstellung möglich, wobei sich der Fluchtpunkt nach eigenem Ermessen festlegen läßt. Selbst die Position der Lichtquelle bei der schattierten Darstellung kann frei bestimmt werden.

Anschließend wird das Objekt je nach eingestelltem Modus (fast alle sind kombinierbar) auf den Bildschirm gezeichnet (Bild 4). Besitzt man einen Drucker, kann die entstandene Grafik auf Papier gebannt werden. Giga-CAD bietet hier aber eine sensationelle Besonderheit. Neben der einfachen Auflösung (320 mal 200 Punkte), wie sie auch auf dem Bildschirm des C64 möglich ist, können die dreidimensionalen Körper auch in vierfacher (640 mal 400 Punkte) oder zehnfacher (1000 mal 640 Punkte!) Auflösung und Größe berechnet und ausgedruckt werden (Bild 5). Eine gedruckte Grafik in zehnfacher Auflösung entspricht dabei einer tatsächlichen Größe eines Bogens Papier im DIN-A3-Format. Der Ausdruck erfolgt aber in Einzelteilen auf DIN-A4-Blättern, die man anschließend miteinander verkleben muß.

Besitzt man keinen Drucker, hält Giga-CAD eine weitere Freude bereit. Aus jedem beliebigen Objekt kann nämlich



ein Film mit insgesamt 24 Bildern erzeugt werden. Giga-CAD berechnet dabei die Ansichtsphasen des sich drehenden Körpers.

Mit Hilfe eines unabängigen Programms werden die einzelnen Phasenbilder von Diskette geladen und schließlich zu einem Film zusammengefügt. Giga-CAD bietet hier unbegrenzte Möglichkeiten.



Bild 4. Der Giga-CAD-Schriftzug als perspektivische Konstruktion mit Schattierung auf dem Bildschirm

Wie Hi-Eddi wird Giga-CAD heute in einer optimierten (schnelleren) und erweiterten Fassung unter dem Namen »Giga-CAD Plus« zusammen mit einem umfangreichen Buch von Markt & Technik angeboten. Das Buch enthält dabei nützliche Tricks und Tips für die Arbeit mit Giga-CAD Plus und beschreibt zusätzlich die neuen Funktionen dieses hervorragenden Programms.

Aber nicht nur auf den Gebieten Floppystation und Grafik wurde Fantastisches geleistet. Auch Computer-Fans, die nach guten Anwender- und Hilfsprogrammen suchten, konnten in den letzten drei Jahren im 64'er-Magazin fündig werden. Besonders für passionierte Freunde der Maschinensprache sind zwei Leser-Listings des 64'er-Magazins von größter Bedeutung.

SMON - Monitor der Superklasse

Wie oft schon hat man einen eingebauten Maschinensprachemonitor im C64 vermißt, mit dem man zumindest Maschinenprogramme laden und speichern kann. Die 64'er-Redaktion war sich dieses Mangels bewußt und veröffentlichte ab Ausgabe 11/84 Schritt für Schritt, ähnlich einem Baukastensystem, den mittlerweile klassisch zu nennenden Monitor »SMON«. Mit dem Erwerb des 64'er-Magazins, Ausgabe 4/85, hatte man schließlich einen kompletten Monitor zur Verfügung, der sich weit über den »Standard« erhob.

Selbstverständlich besitzt der SMON alle Kommandos, die zu einem anständigen Monitor gehören, wie das hexadezimale Auflisten von Speicherbereichen (Memory-Dump), das Laden und Speichern von bestimmten Speicherbereichen, ein Mini-Assembler und ein Disassembler. Der eingebaute Assembler erlaubt zusätzlich die Verwendung von Labels bei der Programmierung in Maschinensprache. Doch schon bei den Befehlen zum Verschieben von Speicherbereichen zeigt SMON seine Fähigkeiten. Neben dem normalen Verschieben kann mittels eines Befehls gleichzeitig ein Angleich von direkten Sprüngen und Adreßzugriffen an den neuen Bereich erfolgen.

Außergewöhnlich interessant ist auch die Vielzahl an Suchroutinen, die das Durchstöbern des Speichers nach verschiedensten Daten erlauben. So kann man nach Maschinenbefehlen suchen, die entweder einen absoluten oder gar einen relativen Operanden haben. Selbst Befehle, die auf Zero-Page-Adressen zugreifen und unmittelbare Adressierung werden von SMON problemlos gefunden. Doch SMON kann auch nach Tabellen suchen. Alle Werte, die nicht als Maschinencode zu identifizieren sind, werden als Tabelle ausgewiesen.

Eine Funktion ist bei der Analyse von Maschinenprogrammen besonders wichtig: der Trace-Modus. Hier wird ein Maschinenprogramm schrittweise abgearbeitet, so daß der Programmmierer Veränderungen in den Prozessor-Registern und im Speicher wahrnehmen kann, um den eventuellen Grund eines Absturzes zu lokalisieren. Anders als in Basic helfen dem Anwender in Masachinensprache keine Fehlermeldungen, um unkorrekte Programmteile zu erkennen. Ein Fehler hat hier meist einen Absturz zur Folge. SMON besitzt aus diesem Grund die unterschiedlichsten Trace-Kommandos.

SMON ist auch Disketten-Monitor

Ohne Absicht betätigen Sie die Taste <Z>, während Sie mit SMON arbeiten. Plötzlich verfärbt sich der Rahmen auf dem Bildschirm, und sämtliche Kommandos, die zuvor noch hervorragend funktionierten, werden nun von SMON nicht mehr akzeptiert. Man darf nun nicht glauben, SMON sei defekt oder abgestürzt. Der komfortable Monitor befindet sich nur in einem anderen Modus. Auf Tastendruck verwandelt sich SMON in einen kleinen Disketten-Monitor, der das Bearbeiten von einzelnen Blöcken einer Diskette ermöglicht. Er ist natürlich bei weitem nicht so umfangreich wie etwa das Disketten-Utility Disc-Wizard, das wir schon vorgestellt haben. SMON beschränkt sich dabei lediglich auf das Lesen, Editieren und Zurückschreiben der Blöcke, leistet aber bei der Arbeit mit dem Floppylaufwerk in Maschinensprache wertvolle Hilfe.

Obwohl SMON einen ungewöhnlich großen Befehlsvorrat hat (wir konnten nur einen Teil anschneiden), ist das Programm selbst sehr kurz. Er benötigt nur etwa 4000 Byte des Speichers Ihres C64 und kann zudem durch einen eigenen Befehl frei verschoben werden, so daß der Monitor immer dort arbeitet, wo er bei der Programmierung in Maschinensprache nicht stört.

Für die Erstellung eigener Maschinenprogramme besitzt SMON einen Assembler, der die direkte Eingabe von Assemblerbefehlen erlaubt und für kleine Programme durchaus ausreichend ist. Größere Maschinensprache-Projekte lassen sich mit ihm jedoch kaum realisieren, da man hier schnell die Übersicht verliert. Für eigene Maschinensprache-Werke sollte man deshalb lieber auf einen großen Assembler zurückgreifen. Besonders empfehlenswert ist dabei ein Leser-Listing aus der Ausgabe 7/85 des 64'er-Magazins (und Sonderheft 8/85):

Mit Hypra-Ass so einfach wie in Basic programmieren

Sein Name ist »Hypra-Ass«. Er darf sich zu Recht einen Assembler der Spitzenklasse nennen, denn er überbietet so manchen teuren, käuflichen Assembler in bezug auf die Leistung um ein Vielfaches.

Wer noch nie in Assembler gearbeitet hat, dem sollen an dieser Stelle einige Erläuterungen zur Maschinensprache gegeben werden: Maschinensprache ist die niederste und

zugleich die einzige Sprache, die ein Computer verstehen kann. Höhere Sprachen müssen deshalb zuvor in diese Maschinensprache übersetzt werden, was durch Interpreter (bei Basic) oder Compiler (zum Beispiel bei Pascal) erfolgt. Maschinensprache hat nun den Vorteil, extrem schnell zu arbeiten, die Programmierung ist jedoch auch recht umständlich und sehr fehleranfällig, da es beispielsweise keine Variablen gibt. Alle Zugriffe müssen direkt auf Speicherstellen des Computers vorgenommen werden. Sprünge sind nur auf Adressen möglich und das Fehlen von Fehlermeldungen sind nur einige Aspekte, die das Programmieren in Maschinensprache zur Qual werden lassen. Um diesen Mängeln abzuhelfen, wurden Programme entwickelt, die das Arbeiten mit Maschinensprache erleichtern: die Assembler. Große Assembler, sogenannte Makro-Assembler bieten dabei bereits sehr angenehme Erleichterungen. Hypra-Ass ist ein solcher Makro-Assembler, dessen Eigenschaften allerding weit über die anderer Programme hinausgehen.

Bild 5. Solche Grafiken können mit »Giga-CAD« auf einem Drucker ausgegeben werden

Es lassen sich, wie in Basic, Variable definieren, die unterschiedliche Werte annehmen können. Was Hypra-Ass von anderen Assemblern unterscheidet, ist die Abgrenzung zwischen globalen und lokalen Variablen, die man eigentlich nur von strukturierten, höheren Programmiersprachen gewöhnt ist. Diese Unterscheidung ist besonders bei einer weiteren Fähigkeit von Hypra-Ass von Bedeutung. Kehren bestimmte Befehlsfolgen in einem Programm immer wieder, so kann man sie unter einem selbstgewählten Namen zusammenfassen. Man nennt diese Gruppierungen auch Makros. Das Besondere ist, daß man ein Makro wie einen normalen Befehl im Programm verwendet, obwohl es ihn eigentlich gar nicht gibt. Tabellen werden in Maschinensprache sehr oft benötigt. Dort stehen beispielsweise wichtige Werte, die für einen Programmteil zur Arbeit erforderlich sind. Mittels einfacher Kommandos lassen sich derartige Tabellen einfach in den Programmtext einfügen. Will man zum Beispiel einen Text in einer Tabelle ablegen, ist es in Maschinensprache normalerweise nötig, jeden einzelnen Buchstaben zunächst in den entsprechenden Code umzuwandeln, um ihn in die betreffende Speicherstelle zu schreiben. Bei Hypra-Ass kann der Text direkt in die Tabelle eingegeben werden, ähnlich den DATA-Statements bei Basic. Die Umwandlung wird beim späteren Übersetzungsvorgang automatisch bewerkstelligt. Hypra-Ass besitzt auch einige Kommandos, die wie in Basic IF, THEN und ELSE genannt werden und ähnliche Funktionen ausführen. Mit ihnen ist es möglich, daß bestimmte Teile des Programmes nur berücksichtigt werden, wenn entsprechende Bedingungen erfüllt sind oder nicht. Maschinensprache-Experten wissen, daß es sich hierbei um »bedingte Assemblierung« handelt.

Für die Eingabe von Assemblerprogrammen hat Gerd Möllmann, der Autor von Hypra-Ass, den Basic-Editor des C64 übernommen und einige Modifikationen angebracht, die ihn noch wesentlich komfortabler gestalten. Die Programmierung mit Hypra-Ass ist also ähnlich einfach wie das Verfassen von Basic-Programmen. Nach Start des Assemblers meldet sich der Computer wieder mit dem üblichen »READY«. Hypra-Ass ist aber schon aktiv. Nun kann man sein Programm mit Zeilennummern eingeben. Alle Basic-Befehle sind dabei im Direkt-Modus noch verwendbar.

Super Editor

Tippt man nun RUN zum Programmstart, wird der eigentliche Assembler aktiviert, der die Übersetzung des Programmes in Maschinensprache vornimmt. Die Übersetzung (Assemblierung) geschieht in drei Durchgängen, deren Arbeitsgeschwindigkeit bemerkenswert hoch ist. Treten während der Assemblierung Ungereimtheiten auf, die Hypra-Ass nicht entziffern kann, gibt er eine seiner zahlreichen Fehlermeldungen aus, die das Auffinden von Fehlern erleichtern.

Die beiden Programme SMON und Hypra-Ass stellen ein hervorragendes Programm-Gespann für den Assembler-Programmierer dar. Dies wurde noch einmal unterstrichen durch die gemeinsame Veröffentlichung beider Programme im Maschinensprache-Sonderheft 8/85, das auch den Einsteiger leicht verständlich an die hohe Kunst der Assembler-Programmierung heranführt.

Der C64 bekommt Rechenunterricht

Vielleicht haben Sie schon leidvoll festgestellt, daß der C64 das Rechnen nicht so genau nimmt, wie man es manchmal wünscht. Falls Sie diesen Vorwurf an den C64 nicht glauben wollen, probieren Sie doch einmal folgendes aus.

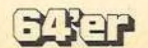
```
PRINT 9↑2
```

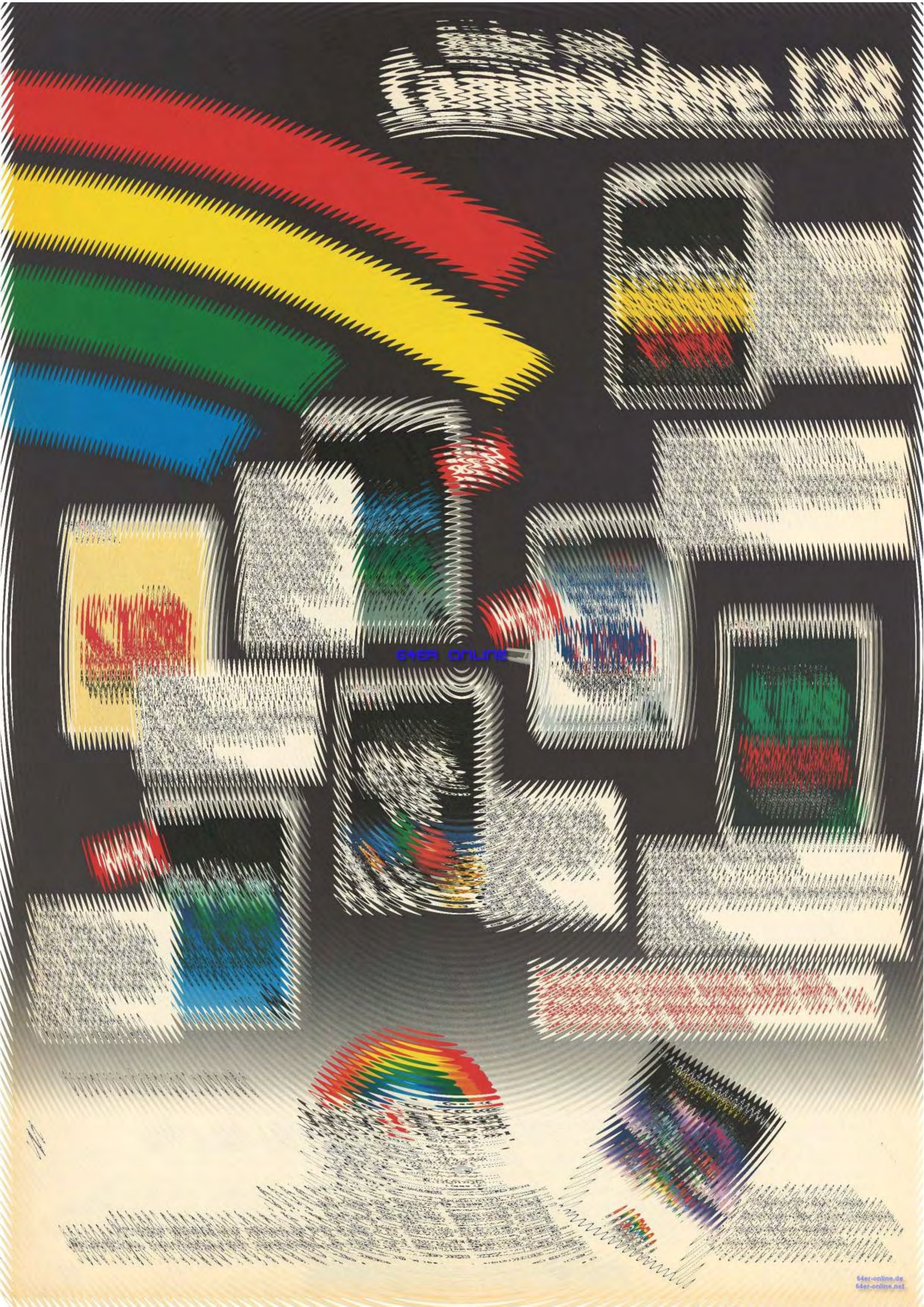
Diese Anweisung müßte eigentlich das Quadrat von 9 errechnen und somit das Ergebnis 81 ausgeben. Der C64 ist jedoch gänzlich anderer Meinung. Er behauptet, das Ergebnis sei 81,0000001.

Dieses Manko beseitigt »Arith13«, unsere Anwendung des Monats aus 64'er 3/87. Denn »Arith13« ist ein außergewöhnliches Programm, das es erlaubt, jegliche Berechnungen auf 13 Nachkommastellen genau vorzunehmen. Gleichzeitig wurden die oben vorgestellten Unzulänglichkeiten des C64-Basic beseitigt, so daß man den C64 zusammen mit dieser Erweiterung zu Recht als Rechengenie bezeichnen kann. Die genauen Rechenroutinen werden mittels eines neuen PRINT-Befehls angesteuert und funktionieren bei allen nur erdenklichen Berechnungen. Selbst die mathematischen Funktionen SIN, COS, TAN und ATN werden mit neuer Genauigkeit errechnet.

Kurvendiskussion perfekt

Nein, es geht hier nicht um Brigitte Bardot oder den Nürburgring. Es geht um die für viele so lästige Mathematik. »Kudi 64«, unsere Anwendung des Monats aus Ausgabe 2/87, bringt dem Anwender die Kurvendiskussion durch





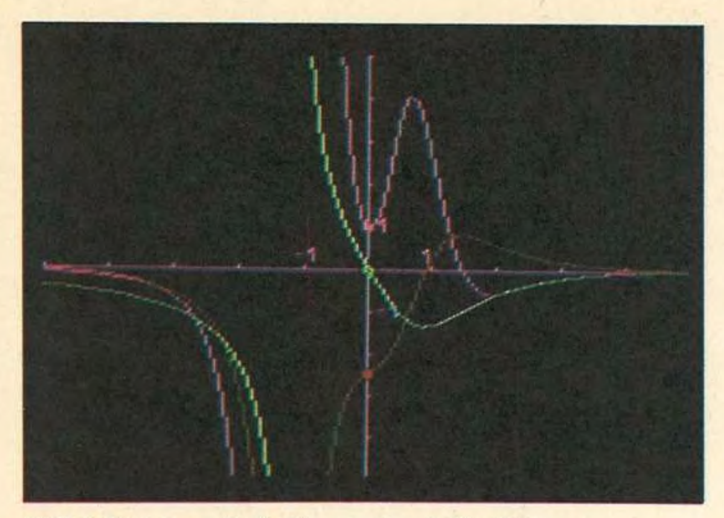


Bild 6. Eine von »Kudi 64« errechnete Funktion

Anschaulichkeit und exakte Berechnung näher. Auch hier wurden, wie bei »Arith13«, optimierte Rechenroutinen verwendet. Kudi 64 ist in der Lage, jede eingegebene Funktion im Grafik-Modus zu zeichnen (Bild 6) und gibt anschließend sowohl die erste und zweite Ableitung als auch Nullstellen und Wendepunkte auf dem Bildschirm aus. Von dem unliebsamen Ballast der Berechnungen befreit, kann man sich als Schüler oder Student mit Kudi 64 voll dem Verständnis für die eigentliche Materie widmen – oder die so gewonnene Zeit für angenehmere Tätigkeiten nutzen.



Bild 7. Die Benutzeroberfläche von »Prodisc«

Ordnung im Diskettenchaos

»Irgendwo muß das Programm doch sein!« Mit diesem Satz und zahlreichen Flüchen auf den Lippen haben Sie sicher schon des öfteren ein wichtiges, dringend benötigtes Programm gesucht und nach dem erfolglosen Sichten von etwa 20 Directories entmutigt aufgegeben. Angesichts dieser Probleme veröffentlichten wir in Ausgabe 6/86 »Prodisc«, ein menügesteuertes Diskettenverwaltungsprogramm. Es arbeitet mit Fenster-Funktionen (Bild 7) und Steuerung per Joystick/Maus und kann pro Datendiskette die Kapazität von über 500 Diskettenseiten oder 1745 Programmnamen verwalten. Es stehen schnelle Sortier- und Suchalgorithmen nach verschiedensten Kriterien zur Auswahl, so daß mit Prodisc das oben beschriebene Problem wohl der Vergangenheit angehört.

Meisterhaft schreiben mit Master-Text

Eine der Hauptanwendungen für Computer ist die Textverarbeitung. Auch um diese Belange hat sich das 64'er-Magazin gekümmert und seinen Lesern mit der Master-Text-Serie zwei vorbildliche Textverarbeitungsprogramme für den C64 und C128 vorgestellt. Zunächst »Master-Text 64«: Dieses in Ausgabe 6 und 7/86 und als verbesserte und überarbeitete Version in Sonderheft 16 veröffentlichte Programm kann schlicht und einfach als Renner bezeichnet werden. Es zeichnet sich durch einfache Bedienung, Menüsteuerung und großen Komfort aus (Bild 8). Als Extras bietet es eine Serienbrief-Funktion und einen 80-Zeichen-Zeigemodus, in dem man einen Eindruck vom Aussehen des später gedruckten Dokuments erhält.

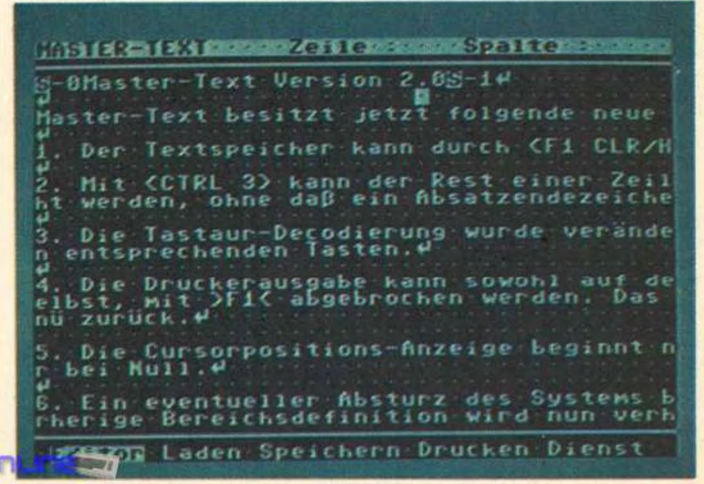


Bild 8. Der Editor von »Master-Text 64«

Master-Text 128, ein Programm für den C128, das in Sonderheft 18 veröffentlicht wurde, darf für sich das Attribut »professionell« beanspruchen. Es bietet neben den Standardoperationen allen Komfort, den man sich als C128-Besitzer wünschen kann. Zum Beispiel: Menü- und Fenster-Technik (Bild 9), Textbausteinfunktionen, eine ständig abrufbare Help-Seite, eine eingebaute Uhr und einen Taschenrechner sowie einen vollwertigen DFÜ-Modus, der auch zum Konvertieren von Texten geeignet ist. Seine Leistungsfähigkeit mußte Master-Text 128 schon in der Testphase beweisen. Mit ihm wurde nämlich eine komplette Diplomarbeit geschrieben, die den Korrektor nicht zuletzt



Bild 9. Traumhafter Komfort mit »Master-Text 128«

wegen ihrer perfekten äußeren Form zu einer sehr guten Bewertung veranlaßte.

Das Ende der Zettelkiste: »Datev«

Ein weiterer Bereich, in dem sich der Einsatz eines Computers geradezu anbietet, ist die Datenverwaltung, beispielsweise bei Adressen, Kundenkarteien, Büchern, Schallplatten etc. Unser Programm »Datev« aus dem Sonderheft 9/86 zum Thema Floppy und Dateiverwaltung wird diesem Zweck in optimaler Weise gerecht. Es ist, wie fast alle der hier vorgestellten Programme, vollständig in Assembler geschrieben und besitzt Funktionen, die man eigentlich nur von professionellen Programmen erwartet. Es ermöglicht die Verwaltung von bis zu 1024 Datensätzen bei einer maximalen Datensatzlänge von 256 Zeichen (also beispielsweise Name, Adresse, Telefon, Beruf, Bemerkung etc.) pro Datendiskette. Ein Datensatz darf dabei bis zu 15 Datenfelder umfassen (Bild 10), von denen bis zu drei als Indexfelder definiert werden können. Nach Begriffen aus diesen Feldern können die Daten später gezielt durchsucht, sortiert oder verglichen werden. Natürlich lassen

```
systemmeldung    479 datensaetze gef.
datensatz-nr.    478 name programme
programm-teil
ausgabe auf >m<onitor oder >d<rucker

PROGRAMM <                          >
DISK-NR. <        >    BLOCKS <     >

Typ      < > (A= Anwender )
             (R= Rollenspiele/Adventure)
             (S= Spiele   )
             (H= Hilfsprg.)
             (X= Sonstige )
 NOTIZ:5
<                                   >
12.Oktober 1985
```

Bild 10. Daten professionell verwaltet mit »Datev«

sich die so gefundenen Daten ausdrucken oder später in einem Textverarbeitungsprogramm für Serienbriefe verwenden. Textverarbeitung und Datenverwaltung arbeiten hier also Hand in Hand.

Der C64 als Klangwunder

Wenn Sie Besitzer einiger Spiele sind, haben Sie sich sicher schon gefragt, wie man diese tollen Musikstücke auf dem C64 schreiben kann, die als Spieluntermalung aus dem Lautsprecher des Monitors kommen. Im krassen Gegensatz dazu haben Ihre Versuche in dieser Richtung nach endlosen PEEK- und POKE-Orgien vielleicht gerade »Hänschen klein« in bescheidener Qualität aus dem Lautsprecher gelockt. Mit dem »Sound-Monitor«, unserem »Listing des Monats« aus Ausgabe 10/86, können auch Sie problemlos professionelle Stücke komponieren (Bild 11). Eine Langspieldiskette (und Hifi-Kassette), die man unter der Bestellnummer MS 630 beim Markt & Technik Verlag bestellen kann, beweist die Leistungsfähigkeit des Sound-Monitors. Viele Stücke auf der Diskette wurden mit diesem Programm komponiert beziehungweise für den C64 umgeschrieben.

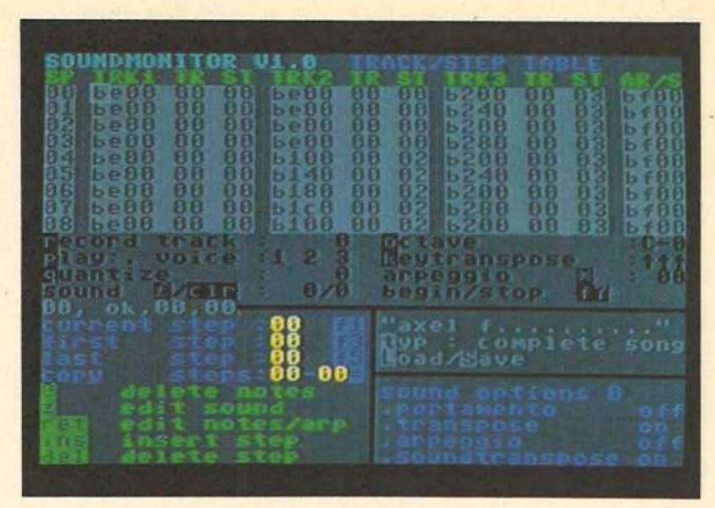


Bild 11. Hitverdächtig: »Der Sound-Monitor«

Mit »Proterm V.6« auf DFÜ-Reise

Die Datenfernübertragung über das Telefon mit Hilfe eines Modems oder Akustikkopplers gewinnt immer mehr an Bedeutung, sowohl im professionellen als auch im Hobbysektor. Mit »Proterm V.6«, dem Listing des Monats aus Ausgabe 4/87, steht Ihnen ein Terminalprogramm zum Abtippen zur Verfügung, das kaum Wünsche offenläßt (Bild 12).

Es bietet neben Standardfunktionen wie der Parametereinstellung auch die Wahl zwischen der Verwendung des ASCII- oder des Commodore-Codes, arbeitet mit Funktionstastenbelegung und stellt zwei Textspeicher zur Verfügung, die getrennt editiert und gesendet werden können. Natürlich kann die Übertragung der Daten auch protokolliert und bei Bedarf auf Diskette gespeichert werden. Sehr nützlich ist auch die Möglichkeit, ganze Programme über das Telefonnetz zu empfangen oder zu senden. Eine wichtige Funktion für den »alten DFÜ-Hasen« stellt die »Autodialer«- und »Autohacker«-Funktion dar. Mit diesen kann Proterm V.6 bei Anschluß eines Modems automatisch Telefonnummern wählen oder auch nach ganz bestimmten Datex-P-Anschlüssen suchen.

Sollten Sie sich also auf die Reise per DFÜ begeben, so wird Ihnen Proterm V.6 ein guter Begleiter sein.

Diese Auswahl aus unseren Spitzen-Programmen ist natürlich längst nicht vollständig. Gerade die Tips, Tricks und Utilities, die im 64'er-Magazin veröffentlicht wurden, sind es, die dem Computer-Freak oder Anwender das Leben mit seinem Hobby leichter machen. Doch schon mit den hier beschriebenen Programmen haben Sie einen Grundstock all dessen, was man zur Arbeit mit dem C64 braucht. (Michael Thomas/sk)

```
Proterm V6.0                    00:01:45
Autohacker
1- Prompt   :login:
   Response :Damien
2- Prompt   :password:
   Response :nope
3- Prompt   :ok?
   Response :ja
4- Prompt   :==>
   Response :dia Boerni
Scanner  <  :000000
Scanner  >  :zzzzzz
Parknua     :45800010006
Timeout     :00:30
Read  Hacker
Write Hacker
```

Bild 12. Mit »Proterm V.6« auf der Reise durchs Telefonnetz



Lernen mit dem C64

Lernen ohne Lehrer – der Computer macht's möglich. Welche nützlichen Programme Sie dabei unterstützen, erfahren Sie hier.

Gegenwärtig befinden wir uns in einer Übergangsphase zu einer Informationsgesellschaft, in der der Computer immer mehr an Bedeutung gewinnt. Dabei interessiert natürlich auch die Frage, wie man den Computer zum schulischen und außerschulischen Lernen nutzen kann. Unternehmen Sie mit uns einen Streifzug durch das mittlerweile schwer zu überschauende Angebot an Lernsoftware. Zunächst ein Überblick, was man alles mit dem Computer lernen kann: Auf spielerische Weise kann man mit dem »Flugsimulator II« seinen Flugschein vorbereiten, oder man beantwortet Quiz-Fragen wie bei »QUIWI« von Kingsoft. Sie können auch Berechnungen durchführen, mathematische Gleichungen und Kurvendiskussionen lösen wie bei »Ali« (Bild 1) von Heureka-Teachware oder »Bruchrechnen« (Bild 2) von Unterrichtsmedien-Hoppius, oder eigenes Briefpapier und Einladungen künstlerisch gestalten mit »Hi-Eddi+« von Markt & Technik. Neue Fachgebiete für den Schulunterricht oder für sich selbst lassen sich zu Hause erarbeiten durch Vokabellernprogramme wie »Learning English – Modern Course« (Bild 3) von Heureka-Teachware, »Polissez Votre Français« von Data Becker oder Rechtschreibung trainieren mit der »Rechtschreibtafel« (Bild 4) von Unterrichtsmedien-Hoppius. Das Zehn-Finger-System ist erlernbar mit »Maschinenschreiben« (Bild 5) aus dem Falken-Verlag, oder Sie wiederholen höhere Mathematik mit »Computerlösungen für Schule und Studium« des MVG-Verlags. Im musikalischen Bereich läßt sich der Computer zum Komponieren einsetzen, beispielsweise mit »Music Shop« von Broderbund oder dem »Sound-Monitor« aus unserer 64'er-Ausgabe 10/86. Auf alle Fälle sollte man mit dem C64 erlernen, Texte mit dem Computer zu verarbeiten. Das erleichtert es, ein Referat zu schreiben, Einladungen und Rundschreiben zu versenden oder Bewerbungen zu verfassen.

Messen, Steuern und Regeln ist ein weiteres Betätigungsfeld, bei dem man mit viel Spaß basteln und eine Menge lernen kann. Viele Versuche aus Physik, Chemie, Biologie und Technik lassen sich durch den Computer hervorragend steuern und auswerten. Versuche, die oft über sehr lange Zeiträume laufen müssen, aber auch blitzschnelle Änderungen kann der C64 erfassen und über den Monitor visuell darstellen. Zum Experimentieren bieten Fischer-Technik und Lego Computer Baukästen für den C64. Beim Arbeitskreis Computer im Chemieunterricht (Institut Dr. Flad, Bretscheidstraße 127, 7000 Stuttgart) erhält man angeblich sogar kostenlos Chemieprogramme mit Begleitheft zum Ausprobieren.

Simulationsprogramme, mit denen man komplexe Systeme und Vorgänge analysieren und simulieren kann, sind im Kommen. Zum Beispiel die CNC-Technologie mit »CNCTIX« vom Klett-Verlag oder Zusammenhänge in unserer Umwelt mit »Umweltdynamik« aus dem TEWI-Verlag. Auch die Datenkommunikation gehört zu den zukunftsträchtigen Anwendungen, die man mit dem C64 erlernen kann. DFÜ und BTX sowie Mailboxen und Datenbanken sind hier die Stichworte. Nun möchten wir Ihnen einige bewährte Programme vorstellen, die mit Schülern und Studenten getestet wurden. Sie sollen aber noch wissen, daß weitere Informationen zu Lernsoftware in den 64'er-Ausgaben 8/86 und 2/87 sowie im Sonderheft 16 (für Einsteiger) zu finden sind.

Die Rechentafel – Bruchrechnen (Bild 2)

Umfang und Handhabung: Alle wichtigen Bereiche der Bruchrechnung sind enthalten – Addition, Subtraktion, Multiplikation, Division, Erweitern und Kürzen. Außerdem werden echte und unechte sowie gemischte Brüche und Kettenaufgaben behandelt. Ein sehr ausführlicher Teil mit Rechenregeln und Beispielaufgaben kann jederzeit angewählt werden, sogar während einer bereits begonnenen Rechenaufgabe. Ein Rücksprung zum Hauptmenü ist möglich.

Didaktische Bewertung: Die Arbeitsweise ist durch gute Menüsteuerung weitgehend selbsterklärend. Die Regelableitungen mit den vorgerechneten Beispielen ist eine der besten Einführungen, die uns bekannt sind. Geschickt ist auch die farbige Darstellung der berechneten Teilschritte während des Lösungsversuchs, wie überhaupt der Bildschirmaufbau als gelungen bezeichnet werden kann.

Die wählbaren Schwierigkeitsgrade sind gut abgestimmt. Leider muß der Bruchstrich jeweils mit »B« extra angefordert werden, was den Arbeitsfluß unnötig hemmt. Das Programm ist mit einem Kopierschutz versehen.

Bild 1. Eine Aufgabe mit dem Algebra-Programm »ALI« mit den einzelnen Rechenschritten

Bild 2. Mit dem Programm »Die Rechentafel« können Sie Bruchrechnen erlernen



Bild 3. Das Programm »Learning English« unterstützt das gleichnamige Buch

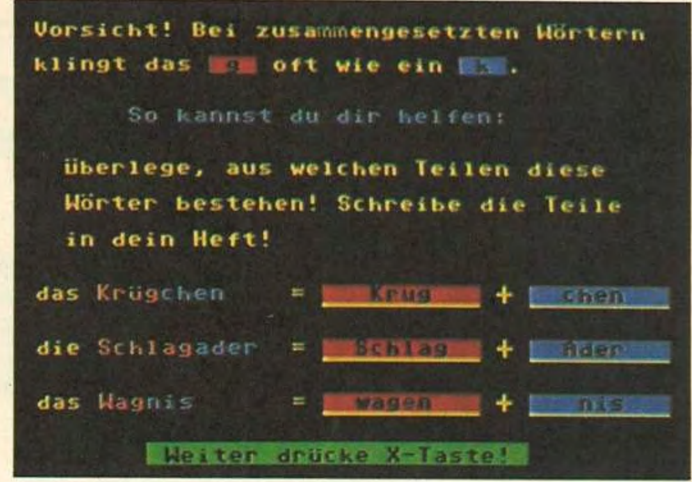


Bild 4. Die »Rechtschreibtafel« hilft den jungen Schülern weiter

Gesamtbewertung: Gut geeignet für Schüler ab Klasse 6 und Auszubildende.

Anbieter und Preis: Unterrichtsmedien-Hoppius, Bannstraße 27, 6330 Wetzlar; Diskette für 79 Mark.

ALI – das intelligente Algebraprogramm (Bild 1)

Umfang und Handhabung: Die neueste Version 4 des mehrfach weiterentwickelten Programms enthält neben binomischen Formeln verschachtelte Klammerberechnungen, quadratische Gleichungen und Potenzen. Faktorenzerlegung und Nullstellenbestimmung höheren Grades sind weitere Schwerpunkte dieses Programms. Implizit definierte Funktionen und grafische Lösungen für Gleichungen mit zwei Unbekannten sind ebenfalls vorhanden.

Die einfache Menüsteuerung gestattet eine bequeme Vorgehensweise, ein Rücksprung ins Hauptmenü ist möglich. Neu ist bei dieser Version die Einbindung eines Floppy-Beschleunigers, der Ladezeiten von nur noch 14 Sekunden ermöglicht. Falls sich wegen schlechter Justierung des Laufwerks Probleme ergeben, kann eine langsamere Laderoutine gewählt werden. Wenn alles nichts hilft, läßt sich diese noch mit POKE 999,9 abschalten und somit auf jeden Fall ein ganz normales Laden mit LOAD"!",8,1 bewirken.

Zusätzlich können Sie Tests erstellen, in denen die Aufgaben als Lückentexte dargestellt sind, eine wesentliche Arbeitserleichterung für Lehrer. Grafische Darstellungen von Funktionen und deren Wertetabellen sind auf dem Drucker darstellbar, wobei durch Voreinstellung der Gerätekonfiguration (Druckertyp und Interface) die Anpassung recht einfach ist.

Didaktische Bewertung: Mustergültig ist das Vorrechnen der Beispielaufgaben. Der jeweilige Teilschritt wird deutlich hervorgehoben. Gibt man eigene Aufgaben ein, so sind persönliche Lösungvorschläge, die vom typischen Rechenweg abweichen, zulässig, wenn diese richtig sind. Falsche Ansätze kommen gar nicht erst auf den Monitor.

Sehr interessant sind auch die Wertetabellen mit den Zeichnungen der zugehörigen Geraden oder Parabeln. Derzeit ist es wohl das beste Programm dieses Fachgebiets für den C64. Die Programmdiskette ist kopiergeschützt.

Gesamtbewertung: Sehr gut geeignet für die Klassen 6 bis 12 an Realschulen, Gymnasien und Berufsfachschulen.

Anbieter und Preise: Heureka-Teachware, Paul-Hösch-Straße 4, 8000 München 60; Diskette 99 Mark. Für 36 Mark erhält man eine Update-Version, wenn man die alte Original-Diskette einsendet.

Geometrie für dritte/vierte Klasse (Bild 6)

Umfang und Handhabung: Das Programm ist auf der Diskette »Bit in Mathematik« enthalten, wobei die Grundrechenarten vom ersten bis vierten Schuljahr behandelt werden. Mit dem dabei enthalten Geometrie-Programm bearbeiten Sie Spiegelbild-Figuren, bei denen die Spiegelachsen bestimmt oder gezeichnet werden müssen. Ferner sind Muster zu zeichnen beziehungsweise fortzusetzen. Würfel und Quader sollen ebenfalls dargestellt werden. Die Eingabe kann über Joystick oder Tasten erfolgen. Eine Rückkehr ins Hauptmenü ist jederzeit möglich.

Didaktische Bewertung: Die Aufgaben, Anweisungen und Menüführungen sind gut gelungen und abwechslungsreich gestaltet. Sie entsprechen der zugehörigen Klassenstufe. Vielleicht wäre es sinnvoller gewesen, das Programm auf Diskette anzubieten, um eigene Entwürfe oder bereits erzielte Teilergebnisse speichern zu können. Obwohl an Motivationshilfen gespart wurde, ist das Programm vom Gesamtkonzept her vernünftig aufgebaut.

Gesamtbewertung: Recht gut geeignet ab (Ende) Klasse 3.

Anbieter und Preise: Georg Westermann Verlag GmbH, Georg-Westermann-Allee 66, 3300 Braunschweig; komplette Diskette für 29,80 Mark

Mathemat

Umfang und Handhabung: Wollte man sämtliche Themen aufzählen, die das Programm beinhaltet, müßte ein eigener, umfangreicher Testbericht darüber geschrieben werden. Daher fassen wir uns kurz. Kurvendiskussion, Vektorrechnung, Geometrie/Algebra, Wahrheitstabellen und Analysis sind die Schwerpunkte. Ein »Mathelexikon« ist auch noch enthalten, ferner einige Druckerroutinen und Diskettenbefehle. Die Menütechnik vereinfacht den Umgang mit dem Programm.

Didaktische Bewertung: Diese Art von Lernsoftware ist nichts für Anfänger. Aber sie eignet sich hervorragend als Ergänzung zum Schulunterricht. Das Programm könnte auch als »Super-Nachhilfestunde« bezeichnet werden. Mit Speeddos-Betriebssystemen läuft das kopiergeschützte Programm nicht.

Gesamtbewertung: Gut geeignet ab Klasse 9 in Realschulen und Gymnasien, teilweise auch für Fachoberschulen.

Anbieter und Preise: Data Becker GmbH, Merowingerstraße 30, 4000 Düsseldorf; Diskette für 19,90 Mark.





Bild 5. Mit »Maschinenschreiben« können Sie das Zehn-Finger-System erlernen

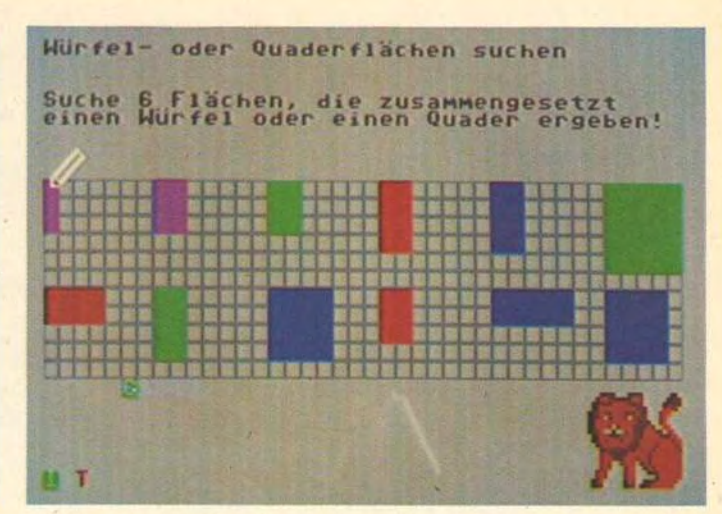
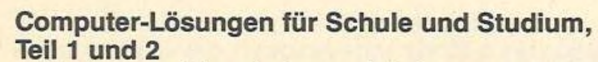


Bild 6. »Geometrie« für die dritte/vierte Klasse wird vom Westermann Verlag angeboten



Computer-Lösungen für Schule und Studium, Teil 1 und 2

Umfang und Handhabung: Auf der ersten Diskette befinden sich schulische Themen ab Klasse 10 wie Zinsrechnung, Primfaktorzerlegung, quadratische Funktionen, lineare Gleichungen, Trigonometrie sowie Differential- und Integralrechnung. Die Handhabung ist unkompliziert.

Ein sehr interessantes Unterprogramm ist auf der ersten Diskette enthalten: ein recht leistungsfähiges Zeichenprogramm mit integrierter Kurvendiskussion.

Die zweite Diskette nimmt die Infinitesimalrechnung wieder auf und führt über Ausgleichs- und Näherungsrechnung, geometrische Muster und Kegelschnitte zu Trigonometrie und analytischer Geometrie. Aufgaben zur Wahrscheinlichkeitsrechnung und Statistik ergänzen das Programmangebot.

Didaktische Bewertung: In Verbindung mit zwei Büchern sind die Herleitungen und algorithmischen Strukturen in pädagogisch angemessener Form dargeboten.

Beide Programme sind nichts für Anfänger, aber zum Erschließen von Lücken, zum Experimentieren und vor allem zur Festigung des Gelernten sind beide Disketten gut einsetzbar.

Sehr nützlich sind die Programme zur dreidimensionalen Darstellung von Funktionen. Didaktisch geschickt aufgemacht ist die Vorgehensweise, wie der Benutzer dazu geführt wird, Körper zu entwerfen und dabei neue Winkel und Koordinatenachsen auszuprobieren. Durch Beispielaufgaben werden beide Programme aufgelockert.

Unerfreulich ist, daß Zahlenreihen nicht untereinander stehen. Die Lesbarkeit wird dadurch erschwert. Ebenso sollten ENETEN statt ENTEN nicht vorkommen. Auf einen Kopierschutz wurde verzichtet.

Gesamtbewertung: Gut geeignete Programme für die gymnasiale Oberstufe, zur Abitur-Vorbereitung und für Studenten.

Anbieter und Preise: mvg-Verlag, Postfach 1761, 8910 Landsberg; Diskette je 58 Mark, Arbeitsbuch Teil 1 und 2 je 29,80 Mark.

Geo: das konstruktive Geometrie-Programm (Bild 7)

Was bietet das Programm im einzelnen?

Es eignet sich zur Unterstützung des Geometrieunterrichts in der Mittelstufe für Lehrkräfte und Schüler von Realschulen und Gymnasien und ermöglicht Achsenspiegelung, Mehrfachbildung und zentrische Streckung.

Diese und andere Grundkonstruktionen wie Streckenübertragungen bis hin zum Thaleskreis sind mit einem einzigen Aufruf durchführbar.

Durch schnelle Eingabe von Standardbezeichnungen in dafür vorgegebene Masken, beispielsweise Großbuchstaben für Punkte oder griechische Buchstaben für Winkel, können alle Konstruktionen rasch erstellt werden, die sonst auf herkömmliche Art mit Zirkel und Lineal gezeichnet werden müßten und oftmals wegen des großen Zeitaufwands im Unterricht verbleiben. Durch übersichtliche Menügestaltung und exakte Hinweise im Begleitheft wird ein schnelles Einarbeiten möglich. Im Falle einer fehlerhaften Eingabe antwortet das Programm zwar mit einem akustischen Signal, jedoch wäre hier wünschenswert, entsprechende Rückfragen als Hilfestellung für den Lernenden einzubinden. Nützlich ist eine Maskenvorgabe als Entscheidungshilfe, die Darstellungsmöglichkeit von vergrößerten Ausschnitten sowie eine individuell wählbare Anpassung auf Interface und Drucker.

Gesamtbewertung: Zur Zeit zählt das Programm »Geo« zu den besten Programmen seines Faches. Geometriekenntnisse werden vorausgesetzt. Das Programm ist deshalb auch als unterrichtsbegleitendes Medium entwickelt worden. Ein Lernfortschritt ist in jedem Falle zu erwarten, zum reinen Selbststudium bedarf es zusätzlicher Fachliteratur. Das Programm hat zwar einen Kopierschutz, doch bietet der Autor für inzwischen defekte Disketten gegen Einsendung der Original-Diskette und eines Unkostenbeitrages von 10 Mark eine neue Diskette an.

Ein sehr schneller Floppy-Beschleuniger ist wie beim Programm »Ali« vorhanden. Insgesamt ist das Preis-Leistungs-Verhältnis sehr gut, das Programm eignet sich gut zur Festigung und Vertiefung der Lerninhalte in den Klassen 7 bis 9, ist aber auch zur Wiederholung in höheren Klassenstufen einsetzbar.

Anbieter und Preise: Heureka-Teachware, Paul-Hösch-Straße 4, 8000 München 60; Diskette mit ausführlichem Begleitheft 64 Mark.

Learning English – Modern Course, Band 1 bis 6 (Bild 3)

Wir haben den sechsten Band bereits in unserer Lernsoftware-Testreihe der Ausgabe 8/86 vorgestellt und beschränken uns hier auf die wichtigsten Punkte. Es ist zu beachten, daß diese Reihe von Peter Ostermann und einem Autorenteam erstellt wurde und nicht mit dem gleichlautenden Programm des Klett-Verlages identisch ist. Der gemeinsame Name bezieht sich nur auf die von Klett herausgegebene Schulbuchreihe.

Das Programm bietet ein lehrbuchbezogenes Vokabeltraining zu dem weitverbreiteten Schulbuch »Learning English – Modern Course« für die Sekundarstufe 1 und ist

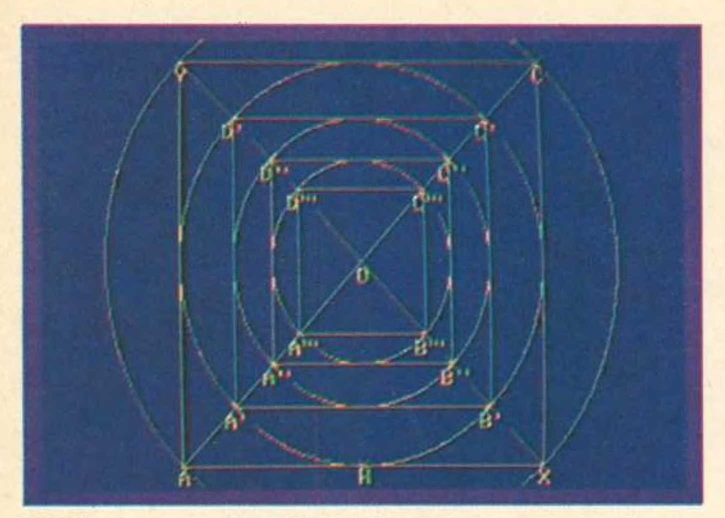


Bild 7. »Geo« von Heureka-Teachware ist hervorragend zur Unterrichtsbegleitung geeignet

Bild 8. Die »Lernkartei« ist ein großes Lernpaket (Grammatik, Erdkunde, Biologie, etc.) für Schüler ab Klasse 4

auch für Wiederholungen in der Oberstufe einsetzbar. Interessant sind die zahlreichen Abfragemöglichkeiten, so daß Langeweile vermieden wird. Man kann vom englischen oder deutschen Wort ausgehen oder von der Definition, besonders wichtig für einsprachiges Lernen! Sowohl in der Reihenfolge der Lektionen als auch kreuz und quer ist ein gezieltes Arbeiten möglich, wobei die Vokabeln nicht nur isoliert, sondern auch im Kontext geübt werden können. Gerade das ist besonders hilfreich, weil man den Stoff, den man im Zusammenhang lernt, viel besser behält. Unregelmäßige Verben können systematisch trainiert werden. Gut ist auch die Lexikonfunktion, die mit der F8-Taste jederzeit aufgerufen werden kann: Unbekannte Vokabeln und ihre Stammformen werden auf den Bildschirm gebracht und erklärt. Dabei kann man vorwärts und rückwärts blättern, sich die Übersetzungen, aber auch die gängigsten englischsprachigen Definitionen ansehen und einprägen.

Schulenglisch per Sofware

Was wir bis jetzt in keinem anderen Vokabellernprogramm fanden: Das Programm erkennt einzelne Rechtschreibfehler und ermöglicht deren Korrektur, sogar bei falschen Präpositionen!

Gesamtbewertung: Die Arbeitsweise ist durch gute Menüsteuerung weitgehend selbsterklärend und jederzeit ein Rücksprung an den Programmanfang möglich. Eine gesonderte Help-Funktion erlaubt gezielte Lösungshilfen. Deutsche Umlaute, Tastaturbelegung nach DIN sowie das »ß« sind selbstverständlich. Das Begleitheft ist klar verständlich geschrieben. Wortschatz und didaktischer Aufbau sind gut. Das Preis-Leistungs-Verhältnis stimmt ebenfalls. Kopierschutz und Floppy-Beschleuniger sind wie bei »ALI« vorhanden. Insgesamt ist das Lernprogramm sehr gut für die Klassen 5 bis 10 einsetzbar.

Anbieter und Preise: Heureka-Teachware, Paul-Hösch-Straße 4, 8000 München 60; Disketten für je 64 Mark.

Maschinenschreiben (Bild 5)

Software zum Erlernen des Zehn-Finger-Blindschreibens werden seit Jahren angeboten, leider sind sie häufig unbrauchbar, weil bei der Entwicklung selten erfahrene Lehrkräfte mitwirken. Erhebliche didaktisch-methodische Fehler sind die Folgen. Was nützt beispielsweise ein Schreibmaschinenprogramm, das nicht einmal die Umlaute »ä«, »ö« und »ü« schreiben kann.

Was bietet nun das Programm »Maschinenschreiben«?

Auf der Systemdiskette befinden sich neben einer Grundlektion noch 32 Übungslektionen, die von einfachen Fingerübungen bis hin zu kompletten Texten reichen. Vor dem eigentlichen Beginn der Übungen geht das gut gegliederte Handbuch auf Sitzhaltung, Arbeitsplatz und Gymnastikübungen zur Lockerung der Finger ein, um dann in verständlicher Sprache die eigentlichen Übungen und Möglichkeiten des Programms darzulegen.

Wahlweise kann man mit deutscher DIN- oder Commodore-Tastatur arbeiten. Als Hilfestellung sind dem Programm Aufkleber beigefügt, die man auf die anders belegten Tasten kleben kann.

Als äußerst nützlich erweisen sich die Optionen »Fehlergeräusch« und »Metronom« (Taktgeber), die variabel einstellbar sind und der Einübung eines gleichmäßigen Anschlags sowie der Fehlerrückmeldung dienen. Wen das anfangs stört, der kann beide Funktionen auch abschalten. Die Schreibgeschwindigkeiten lassen sich über die Tasten <+> bzw. <-> (schneller/langsamer) bestimmen.

Neben der Möglichkeit, mit den Übungen auf der Programmdiskette zu arbeiten, gibt es auch über eine bequeme Menüführung die Option, eigene Texte einzugeben und diese zu speichern. Dies ist besonders sinnvoll, wenn man mit einem bestimmten Lehrgang, beispielsweise für Abendkurse, zielgerichtet üben möchte. Komfort bieten auch die Funktionen »Directory« und »Texte löschen«.

Gesamtbewertung: Über ein kleines Textverarbeitungsprogramm lassen sich die Texte auch ausdrucken. Allerdings läuft das Druckprogramm anscheinend nur korrekt bei Direktanschluß über Centronics-Schnittstelle oder mit Commodore-Druckern.

Wer möchte, kann sogar die fertigen Lektionen editieren – das fanden wir bisher nirgends. Ein Ergebnisprotokoll macht vernünftige Angaben zum Lernfortschritt.

Das Programm eignet sich gut für alle, die aus beruflichen oder privaten Gründen rasch das Zehn-Finger-System erlernen wollen.

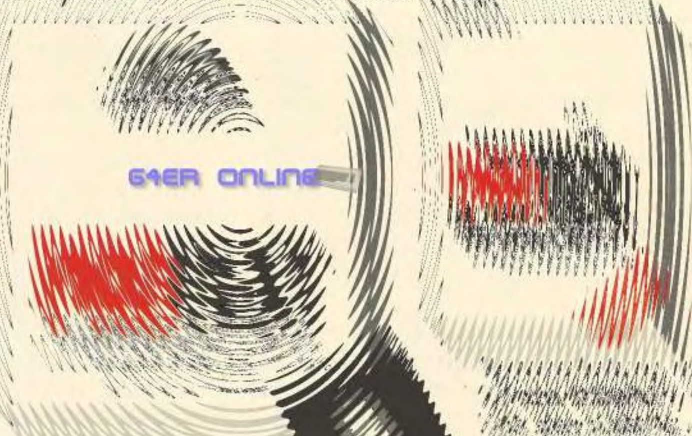
Anbieter und Preise: Unterrichtsmedien-Hoppius (siehe Rechentafel) und Falken-Verlag, Postfach 1120, 6272 Niedernhausen/Ts; Diskette 49,80 Mark.

Rechtschreiben mit Köpfchen, Teil 1 (Bild 4)

Umfang und Handhabung: Ein Rechtschreibtrainingsprogramm mit Bildern und Melodien; es besteht aus drei Disketten und schriftlichem Begleitmaterial für ergänzende Übungen. In Teil 1 werden ähnlich klingende Endlaute bei Substantiven trainiert, beispielsweise Berg/Werk oder




PROTEXT





Rad/Tat. Die zweite Diskette behandelt Endlautverhärtungen wie singt/sinkt. Der Teil 3 unterscheidet Endlaute bei Adjektiven wie rund/bunt und bearbeitet die Rechtschreibprobleme, die bei zusammengesetzten Wörtern auftreten können, beispielsweise in Windrad.

Didaktische Bewertung: Das Programm enthält überzeugende Beispiele und gute Kombination von Regelableitung mit Aufgabenteil. Gut sind auch die differenzierten Übungen mit selbst wählbarem Schwierigkeitsgrad; der Wortschatz ist akzeptabel. Positiv auf die Motivation dürfte sich gerade bei jüngeren Schülern das abwechslungsreiche Bildmaterial mit den Melodien auswirken. Begleitheft und Arbeitsmappe sind gut einsetzbar.

Gesamtbewertung: Sehr gut geeignet als Lese- und Rechtschreibtraining ab (Ende) Klasse 2 bis etwa Klasse 6.

Anbieter und Preise: Ernst-Klett-Verlag, Postfach 809, 7000 Stuttgart; sowie Unterrichtsmedien - Hoppius, Bannstraße 27, 6330 Wetzlar; je Diskette 49 Mark.

Lernkartei (Bild 8)

Einem Schweizer Mittelstufenlehrer arbeitete die Schulbürokratie viel zu langsam, und so wurde aus seiner Klasse das erste schuleigene EDV-Zentrum der Schweiz. Über ein Multi-User-System mit Floppy 1541 werden dort sechs C64-Computer gesteuert. Seine Schüler zeigten sich ebenso experimentierfreudig, und herausgekommen ist eine randvolle Diskette mit dem Titel »Lernkartei«.

Was bietet die Lernkartei?

Sie ist nicht bloß ein Trainingsprogramm für Rechtschreibung und Grammatik, sie enthält auch Erdkunde, Biologie etc. und liefert außerdem ein Eingabeprogramm zur Erstellung eigener Lerndateien. Noch eines ist hervorzuheben: Man kann aus den Dateien einzelne Fragen zusammensuchen und damit einen Prüfungsbogen erstellen. Wem das immer noch nicht reicht, der kann sich ein Arbeitsblatt ausdrucken lassen und mit dem auf dem Monitor unsichtbaren Codewort »schubi« das Lösungsblatt dazuheften.

Gesamtbewertung: Die Arbeitsweise ist durch gute Menüsteuerung weitgehend selbsterklärend. Ein Rücksprung zum Hauptmenü ist fast immer möglich. Umlaute sowie das »ß« sind vorhanden. Das Begleithandbuch ist klar verständlich geschrieben, die zahlreichen Beispiele erleichtern den Umgang mit diesem Programm. Der Wortschatz und die Grammatikübungen sind noch nicht in allen Fällen an den bei uns verbindlichen Standardwortschatz angeglichen, um als typisches Schulprogramm zu gelten. Die auf deutsche Verhältnisse zugeschnittenen Übungsdisketten sollen jedoch bald erhältlich sein. Angesichts der vielfältigen Möglichkeiten ist das Preis-Leistungs-Verhältnis akzeptabel. Die »Lernkartei« ist ab Klasse 4 gut einsetzbar.

Anbieter und Preise: Huesmann+Benz Verlag, Hochwaldstraße 18, 7700 Singen; Diskette für 90 Mark.

Softlearning - Grundkurse in Englisch, Französisch, Spanisch und Italienisch; Intensivkurse in Englisch, Französisch, Spanisch, Italienisch, Schwedisch und Russisch

Zum Abschluß noch eine völlig neue Konzeption: Zu dieser Art des Lernens, das von dem bulgarischen Lernforscher Prof. Lozanov anwendungsreif entwickelt wurde, sind einige Anmerkungen erforderlich.

In zahlreichen Anzeigen und wissenschaftlichen Beiträgen ist diese gänzlich andere Lernmethode unter dem Begriff »Superlearning« bekannt geworden. Was ist nun das Neue an dieser Art des Lernens?

Wie Sie sicher wissen, lernt man schon vor der Geburt ununterbrochen freiwillig und sozusagen automatisch, um zu überleben. Wir alle, und sämtliche Kinder aller Nationen dieser Erde, erlernen daher so schnell und so einfach unsere Muttersprache, weil wir ja nur auf diesem Wege unsere Wünsche besser artikulieren können und deren Erfüllung erreichen. Ohne Kommunikation zu den Mitmenschen, insbesondere zu den Eltern, müßten wir scheitern.

Später in der Schule sieht das leider ganz anders aus, denn das natürliche, fast spielerische Lernen des Kindes ist nicht mehr gefragt. Ganz im Gegenteil! Der schulische Drill läuft im Prinzip darauf hinaus, daß alle Schüler zur gleichen Zeit denselben Stoff lernen. Widerstände und Hmemmschwellen sind die Folge.

Superlearning will diese vermeiden und beginnt deshalb mit einer Tiefentspannungs-Übung. Dieses Lockerungstraining, auch als »Alpha-Zustand« bezeichnet, beginnt mit pulsierender Grafik und Musik, begleitet von einem Atemtraining.

Zusätzliche Muskelentspannungs-Übungen (wie autogenes Training) verstärken die Alpha-Phase.

Erst danach erfolgen die eigentlichen Sprachkurse, die nach didaktischen Gesichtspunkten gestaltet sind. Gewissermaßen am Bewußtsein vorbei werden die Lerninhalte direkt in das Langzeitgedächtnis transportiert.

Lernen mit Gefühl

Die Werbung arbeitet übrigens schon seit Jahren so! Harmonie und Gefühle werden günstig beeinflußt, und unsere persönlichen Konflikte und Alltagsprobleme sollen auf diesem Wege ausgeklammert werden.

Das Superlearning nach Lozanov ist eine erfolgversprechende, neue Lernmethode, die im Management schon längst praktiziert wird.

Umfang und Handhabung: Neben dem C64 mit Diskettenlaufwerk und Monitor braucht man noch einen Kassettenrecorder, der mit Audio- und Fernbedienungsbuchse ausgerüstet sein muß.

Zu jedem Kurs werden bis zu 4 Sprach-Kassetten geliefert, die über die sogenannte »Systembasis« mit dem Computer synchronisiert werden. Diese Systembasis ist die Grundlage für alle Softlearningkurse und braucht nur einmal angeschafft zu werden.

Nach Erreichen des Alpha-Zustandes beginnt man mit den eigentlichen Sprachübungen. Die Dialogtexte werden vorgesprochen und synchron auf dem Monitor dargestellt. Silbenrätsel, Lückentexte, Multiple-Choice-Aufgaben bei denen man unter mehreren Antworten auswählen kann und Grammatik sind regelmäßig wiederkehrende Übungsformen.

Bis man sich in die einzelnen Vorgehensweisen eingearbeitet und die diversen Kabel richtig angeschlossen hat, vergeht eine ganze Weile.

Didaktische Bewertung: Das Konzept ist grundsätzlich gut, die Aufmachung und der Wortschatz sind dem Lernniveau von Erwachsenen angepaßt. Nicht unwichtig ist es für den Lernerfolg, daß man an die Methode »glaubt«, sonst erreicht man nie die vorhin beschriebene Tiefentspannung, die ja Voraussetzung für diese besondere Lernmethode ist.

Beim Lernen der Vokabeln hilft die Methode der sogenannten »Keywords« (Eselsbrücken). Manche Keywords sind allerdings weit hergeholt und entsprechen kaum der Wirklichkeit. Ladezeiten können inzwischen irritieren.

Gesamtbewertung: Das Lernverfahren ist gut geeignet ab Klasse 13 und für Erwachsene.

Anbieter und Preise: SM-Softtraining GmbH, Schwanthalerstraße 60, 8000 München 2; Disketten für C64: Grundkurse je 199 Mark, Intensivkurse je 99 Mark, Systembasis mit Steuermodul 89 Mark. (Rüdiger Werner/kn)

Der springende Punkt

Roboter sind wirklich nützliche Geräte. In diesem tollen Spiele-Listing soll ein hüpfendes Roboter-Ei eine Nachricht überbringen. Stellen Sie sich der Herausforderung, es ins Ziel zu bringen.

Die Hauptperson dieses Spiels für den C64 ist ein kleiner Roboter, namens »Roboegg«. Vor ihm liegt ein Labyrinth mit Aufzügen, Förderbändern und Statuen (Bild 1). Doch es warten viele Gefahren auf den springlebendigen Roboter, der durch den Joystick in Port 2 gesteuert wird. Das Ziel jedes Levels ist, lebend das »EXIT«-Feld zu erreichen. Dabei läuft keine Uhr mit, und es gibt auch keine Punktewertung. Roboegg hat drei Leben. Wenn er zweimal auf der Stelle springt, beginnt er nach dem vorzeitigen Ableben an dieser Stelle wieder. Ein Druck des Joysticks nach hinten, und es geht weiter. Durch den Feuerknopf kommt man in das Hauptmenü zurück.

Das Hauptmenü hat fünf Punkte, die man durch die entsprechende Zahlentaste anwählt:
1. Spielen 2. Editieren 3. Laden 4. Saven 5. Ende

In der abgedruckten Version befindet sich ein kurzer Demo-Level, der Ihnen die Auswirkung der verschiedenen Spieleelemente zeigt. Der Baum, die Figur, das Kreuz, die Förderbänder und die Pyramide haben keine schädlichen Nebenwirkungen, aber die Dolche, Sträucher und Barrieren. Nachdem Sie den eingebauten Level beherrschen, können Sie sich ans Gestalten Ihrer eigenen Spielfelder machen. Im Editor wird der Cursor nicht durch den Joystick gesteuert, sondern über die Cursor-Tasten. Durch die Taste <X> setzen Sie ein Zeichen, das Sie durch <Z> gewählt haben. Bildteile können entweder aus dem Bild herausgenommen werden oder aus dem Zeichensatz-Menü. Um dieses auf den Bildschirm zu zaubern, drückt man einfach die SPACE-Taste und in der Mitte öffnet sich ein Window, das alle Zeichen enthält. Es verschwindet wieder, wenn man den Bildschirm etwas zur Seite scrollt. Die Bildteile unter dem Window werden dabei nicht gelöscht. Wenn Sie aber den Bildschirm löschen wollen, brauchen Sie nur <SHIFT CLR/HOME> zu drücken.

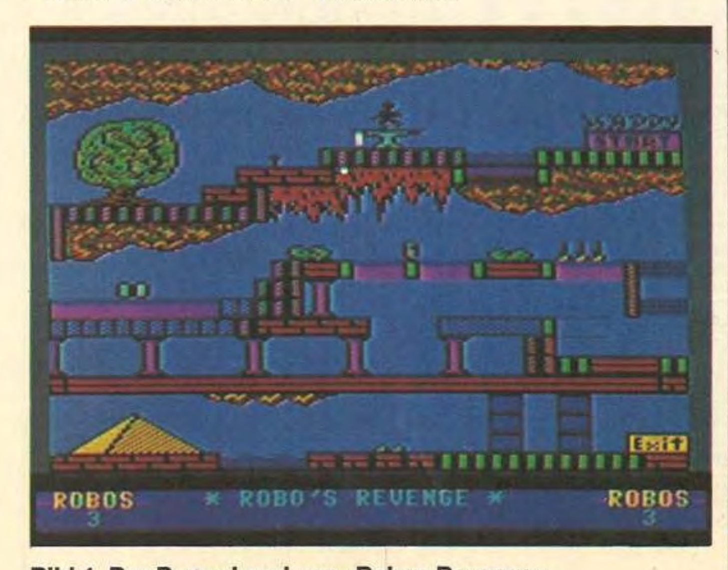


Bild 1. Der Demo-Level von »Robos Revenge«

Der Editor hat mehr zu bieten, als nur ein gewöhnliches Construction Set. Durch den Linkspfeil schalten Sie nämlich im Spielfeld-Editor den Zeichensatz-Editor ein. So sind Sie nicht auf die vorhandenen Zeichen beschränkt, sondern können sich auch eigene Muster einfallen lassen. Die einzige Einschränkung ist die Wirkung der Hindernisse. Auch werden sich die veränderten Zeichen weiterhin bewegen, wenn sie schon in der Urform als bewegte Zeichen gestaltet waren.

Eigene Spielfelder

Wenn Sie anfangen, mit dem Zeichensatz-Editor zu arbeiten, erscheint links oben das bekannte Window mit allen Zeichen. Darin können Sie den Zeichensatz-Cursor bewegen. Rechts oben ist das Zeichen unter dem Cursor vergrößert dargestellt. Wenn Sie die SPACE-Taste drücken, wechseln Sie zum Editor-Cursor und können das Zeichen verändern. Für den Zeichensatz stehen Ihnen drei Grundfarben zur Verfügung, und in jedem Zeichen haben Sie eine frei wählbare Farbe. Durch die Tasten <1> bis <4> setzt man ein farbiges Pixel. Um die Farbe, die Sie durch diese Taste erreichen, zu ändern, drücken Sie gleichzeitig die Commodore-Taste und die Zahlentaste. Beachten Sie bitte, daß Sie nur den Farbwert der Taste <3> für jedes Zeichen beliebig ändern dürfen. Mit der SPACE-Taste schaltet man zwischen dem Zeichensatz-Cursor und dem Editier-Cursor um. Durch <RUN/STOP> verlassen Sie den Zeichensatz-Editor und kehren zum Spielefeld-Editor zurück. Um ins Hauptmenü zurückzukehren, drücken Sie nochmals die RUN/STOP-Taste.

Bei der Funktion »Saven« erscheint keine Meldung oder Abfrage. Sie befinden sich trotzdem in der Speicherfunktion! Drücken Sie jetzt eine Zahlentaste (0 bis 9), und Ihr Spielfeld wird mit dieser Kennung gespeichert. Um ein Spielfeld zu laden, gehen Sie im Menüpunkt »3« genauso vor. Wollen Sie jedoch die Funktion Laden und Speichern nicht ausführen, drücken Sie nur eine Buchstaben-Taste.

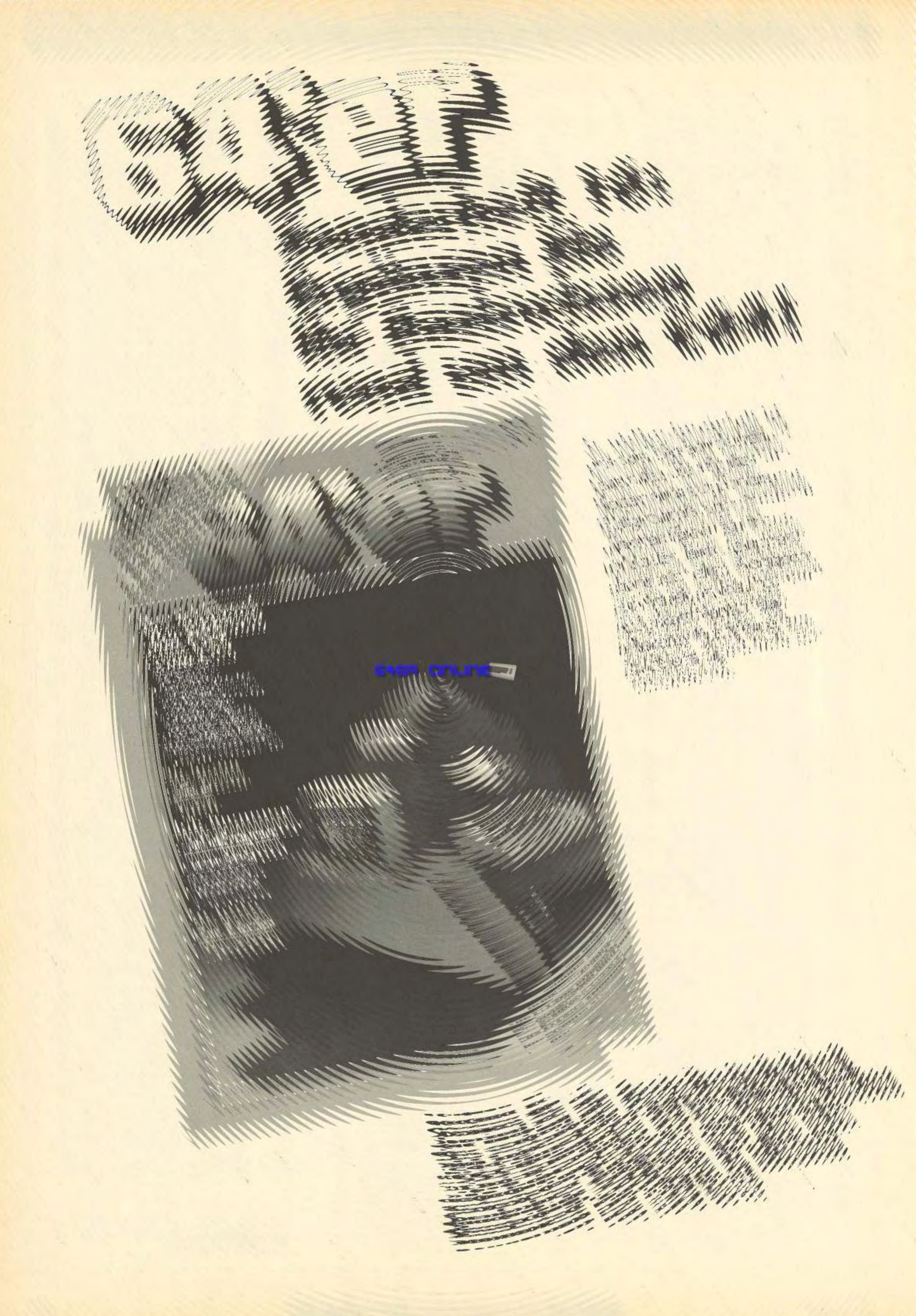
(R. Brandl/T. Schmidt/jk)

```
Name : robo's revenge     0801 2406
-----------------------------------

0801 : 1b 08 00 00 9e 28 32 30  75
0809 : 38 32 29 20 48 41 50 50  19
0811 : 59 20 43 4f 4d 50 55 54  8a
0819 : 45 52 00 00 00 20 20 a2  4e
0821 : e9 bd 32 08 9d ff 00 ca  e6
0829 : d0 f7 78 86 01 ca 9a 4c  4d
0831 : 00 01 a0 00 84 fd 84 fe  22
0839 : a9 06 a2 24 85 ae 86 af  5a
0841 : a9 1c a2 09 85 ac 86 ad  f5
0849 : 20 81 01 a9 01 a2 08 85  f0
0851 : ae 86 af b1 fd c9 36 d0  0d
0859 : 10 a9 00 48 20 a2 01 aa  b7
0861 : 68 20 a8 01 ca d0 fa f0  25
0869 : 0d c9 a4 d0 06 20 a2 01  8c
0871 : 48 d0 e9 20 a8 01 20 b1  16
0879 : 01 d0 d8 ae b8 01 ca 30  0d
0881 : 24 bd b9 01 85 fe 85 2e  d5
0889 : ca bd b9 01 85 fd 85 2d  79
0891 : ca bd b9 01 85 ad ca bd  35
0899 : b9 01 85 ac 8e b8 01 20  bd
08a1 : 81 01 4c 49 01 a9 37 85  24
08a9 : 01 29 06 c9 06 d0 01 58  95
08b1 : 4c 53 c8 a5 fd d0 02 c6  8a
08b9 : fe c6 fd a5 ae d0 02 c6  55
08c1 : af c6 ae b1 ae 91 fd a5  70
08c9 : ae c5 ac d0 e6 a5 af c5  85
08d1 : ad d0 e0 60 20 b1 01 b1  21
08d9 : fd 60 91 ae e6 ae d0 02  6c
08e1 : e6 af 60 e6 fd d0 02 e6  d0
08e9 : fe 60 20 01 08 2b 13 2c  be
08f1 : 0b c0 33 ec 0e 03 9f ef  1e
08f9 : 6d b6 c6 a5 74 42 c8 e7  74
0901 : 75 99 cc 30 7a 01 cf 8e  88
0909 : 7c 00 04 03 03 03 03 03  41
0911 : 03 03 00 a9 00 20 71 a8  e3
0919 : 4c ae a7 a9 ff 8d 6f 03  0b
0921 : 20 50 c8 a9 d8 8d 16 d0  c5
0929 : a9 18 8d 18 d0 a2 03 8e  90
0931 : 15 d0 ca 8e 10 d0 8e 1d  2f
0939 : d0 ca ca 8e 3c 03 a9 1f  b3
0941 : 8d 00 d0 a9 39 8d 01 d0  dd
0949 : 8d 03 d0 a2 0e 8e 02 d0  df
0951 : e8 8e f8 07 8e f9 07 a9  c7
0959 : 04 85 f4 a9 29 85 f3 a9  74
0961 : 21 85 07 a9 80 85 06 a2  cd
0969 : 07 a9 00 9d 00 20 a9 55  4b
0971 : 9d 08 20 a9 aa 9d 10 20  68
0979 : a9 ff 9d 18 20 ca 10 e9  f9
0981 : e8 8e 09 13 8e 0a 13 4c  73
0989 : 96 11 20 ce 11 a9 00 85  f3
0991 : c6 a8 ee 27 d0 ee 28 d0  13
0999 : c5 c6 f0 f6 ad 77 02 d9  2f
09a1 : c9 12 f0 0a c8 c8 c8 c0  68
09a9 : 39 d0 f4 4c 6f 10 b9 ca  05
09b1 : 12 85 f7 b9 cb 12 85 f8  10
09b9 : 6c f7 00 ad 3c 03 d0 21  38
09c1 : ad 00 d0 18 69 08 c9 9f  e3
09c9 : f0 14 8d 00 d0 e6 f3 d0  dc
09d1 : 02 e6 f4 a5 06 18 69 08  0f
09d9 : 85 06 90 02 e6 07 4c 6f  7c
09e1 : 10 ad 02 d0 18 69 10 c9  03
09e9 : 4e f0 f3 8d 02 d0 ee 09  d2
09f1 : 13 4c c3 10 ad 3c 03 d0  88
09f9 : 23 ad 00 d0 38 e9 08 c9  93
0a01 : 17 f0 16 8d 00 d0 a5 f3  cd
0a09 : d0 02 c6 f4 c6 f3 a5 06  d9
0a11 : 38 e9 08 85 06 b0 02 c6  6c
0a19 : 07 4c 6f 10 ad 02 d0 38  c3
0a21 : e9 10 c9 fe f0 f3 8d 02  4d
0a29 : d0 ce 09 13 4c fe 10 ad  5d
0a31 : 3c 03 d0 26 ad 01 d0 18  3e
0a39 : 69 08 c9 a1 f0 19 8d 01  5d
0a41 : d0 a5 f3 18 69 28 85 f3  ba
0a49 : 90 02 e6 f4 a5 06 18 69  f0
0a51 : 80 85 06 90 02 e6 07 4c  34
0a59 : 6f 10 ad 03 d0 18 69 08  20
0a61 : c9 79 f0 f3 8d [illegible] [illegible] ee  b3
0a69 : 0a 13 4c 3c 11 ad 3c 03  0d
0a71 : d0 26 ad 01 d0 38 e9 08  66
0a79 : c9 31 f0 19 8d 01 d0 a5  aa
0a81 : f3 38 e9 28 85 f3 b0 02  ce
0a89 : c6 f4 a5 06 38 e9 80 85  d3
0a91 : 06 b0 02 c6 07 4c 6f 10  f9
0a99 : ad 03 d0 38 e9 08 c9 31  6b
0aa1 : f0 f3 8d 03 d0 ce 0a 13  20
0aa9 : 4c 7a 11 a9 00 8d 6f 03  dc
0ab1 : 60 a9 29 85 f7 85 f9 a9  c8
0ab9 : 04 85 f8 a9 d8 85 fa 20  d9
0ac1 : a6 cc 4c 6f 10 a0 07 a5  3c
0ac9 : 06 85 f7 a5 07 18 69 77  0a
0ad1 : 85 f8 a9 00 91 06 91 f7  bc
0ad9 : 88 10 f9 4c 6f 10 a9 ff  8f
0ae1 : 4d 3c 03 8d 3c 03 4c 6f  ab
0ae9 : 10 a0 00 b1 f3 aa bd 00  0b
0af1 : 96 8d 07 13 a2 00 b1 06  6f
0af9 : 48 29 03 9d 0e 13 68 4a  fa
0b01 : 4a 48 29 03 9d 0d 13 68  79
0b09 : 4a 4a 48 29 03 9d 0c 13  23
0b11 : 68 4a 4a 29 03 9d 0b 13  c6
0b19 : a4 e8 04 c8 c0 08 d0 d5  87
0b21 : a2 00 a0 00 a9 47 85 d1  7a
0b29 : 85 fb a9 04 85 d2 a9 d8  de
0b31 : 85 fc a9 04 8d 06 13 bd  f0
0b39 : 0b 13 91 d1 ad 07 13 91  ef
0b41 : fb c8 bd 0b 13 91 d1 ad  d1
0b49 : 07 13 91 fb c8 e8 ce 06  d9
0b51 : 13 d0 e4 a5 d1 18 69 20  7e
0b59 : 85 d1 85 fb 90 04 e6 d2  12
0b61 : e6 fc e0 20 d0 cc 60 ee  d4
0b69 : 22 d0 ee 01 9f 4c 6f 10  09
0b71 : ee 23 d0 ee 02 9f 4c 6f  30
0b79 : 10 a2 00 a1 f3 aa bd 00  9a
0b81 : 96 18 69 01 09 08 9d 00  e5
0b89 : 96 4c 96 11 ee 21 d0 ee  26
0b91 : 00 9f 4c 6f 10 a9 55 8d  21
0b99 : 08 13 4c 9b 12 a9 aa 8d  e6
0ba1 : 08 13 4c 9b 12 a9 ff 8d  43
0ba9 : 08 13 4c 9b 12 a9 00 8d  4b
0bb1 : 08 13 4c 9b 12 a5 06 85  3b
0bb9 : f7 a5 07 18 69 77 85 f8  a2
0bc1 : ae 09 13 bd 02 13 2d 08  ee
0bc9 : 13 8d 08 13 bd 02 13 49  d2
0bd1 : ff ac 0a 13 31 06 0d 08  93
0bd9 : 13 91 06 91 f7 20 ce 11  46
0be1 : 4c 6f 10 1d a0 10 9d d9  41
0be9 : 10 91 52 11 11 14 11 03  75
0bf1 : 90 11 20 c3 11 93 aa 11  05
0bf9 : 21 7b 12 22 83 12 23 8b  0d
0c01 : 12 24 93 12 31 7b 12 32  e8
0c09 : 83 12 33 8b 12 34 93 12  09
0c11 : 81 4d 12 95 56 12 96 5f  7f
0c19 : 12 97 72 12 c0 30 0c 03  99
0c21 : 36 34 3c 00 00 eb 00 00  e0
0c29 : eb 00 00 eb 00 00 3c 36  ef
0c31 : 23 c3 36 12 3c 00 00 eb  a1
0c39 : 00 00 eb 00 00 eb 00 00  94
0c41 : 3c 36 20 c3 36 15 3c 00  16
0c49 : 00 eb 00 00 eb 00 00 eb  d6
0c51 : 00 00 3c 36 1d c3 36 18  20
0c59 : 3c 00 00 eb 00 00 eb 00  c3
0c61 : 00 eb 00 00 3c 36 1a c3  bc
0c69 : 36 1b 3c 00 00 eb 00 00  9b
0c71 : eb 00 00 eb 00 00 3c 36  37
0c79 : 17 c3 36 1e 3c 00 00 eb  5f
0c81 : 00 00 eb 00 00 eb 00 00  dc
0c89 : 3c 36 14 c3 36 21 3c 00  bb
0c91 : 00 eb 00 00 eb 00 00 eb  1e
0c99 : 00 00 3c 36 11 c3 36 24  c0
0ca1 : 3c 00 00 eb 00 00 eb 00  0b
0ca9 : 00 eb 00 00 3c 36 0e c3  d4
0cb1 : 36 27 3c 00 00 eb 00 00  e9
0cb9 : eb 00 00 eb 00 00 3c 36  7f
0cc1 : 0b c3 36 2a 3c 00 00 eb  1c
0cc9 : 00 00 eb 00 00 eb 00 00  24
0cd1 : 3c 36 08 c3 36 2d 3c 00  61
0cd9 : 00 eb 00 00 eb 00 00 eb  66
0ce1 : 00 00 3c 36 05 c3 36 30  5f
0ce9 : 3c 00 00 eb 00 00 eb 00  53
0cf1 : 00 eb 00 00 3c 00 00 c3  32
0cf9 : 36 33 3c 00 00 eb 00 00  37
0d01 : eb 00 00 3c 00 00 c3 36  ef
0d09 : 36 3c 00 00 eb 00 00 3c  95
0d11 : 00 00 c3 36 39 3c 00 00  3e
0d19 : ff 00 00 c3 a4 20 00 a4  25
0d21 : 20 00 a4 20 00 a4 20 aa  69
0d29 : b7 9a 96 98 99 95 98 98  c0
0d31 : 99 95 95 98 99 95 99 98  eb
0d39 : 9a 98 95 96 95 98 9a 99  13
0d41 : 98 9b 9c 9d 9e a5 a6 a7  83
0d49 : a8 aa ac 93 94 96 9a 98  7d
0d51 : 99 9b b7 a4 20 00 a4 20  0f
0d59 : 00 a4 20 00 a4 20 bd c7  85
0d61 : 99 96 9a 96 9a 98 96 98  b9
0d69 : 99 96 99 9b 9c 9d 9e 93  7f
0d71 : 94 96 99 9a 99 9b 9d 9e  34
0d79 : a4 20 0c a5 a8 aa ac 20  b8
0d81 : c7 a4 20 00 a4 20 00 a4  37
0d89 : 20 00 a4 20 bd c7 96 98  7c
0d91 : 99 98 9b 9c 9d 9e a5 a6  a3
0d99 : a7 a4 20 06 a5 a6 a7 ac  e3
0da1 : 20 20 32 a4 20 11 c7 a4  e5
0da9 : 20 00 a4 20 00 a4 20 00  9c
0db1 : a4 20 bd c7 9b 9c 9d 9e  20
0db9 : 20 b1 b2 b3 b4 b5 b6 a4  f2
0dc1 : 20 0b 41 42 43 a4 20 0a  ed
0dc9 : ea eb ec ed ed ef c7 a4  68
0dd1 : 20 00 a4 20 00 a4 20 00  c4
0dd9 : a4 20 bd c7 a4 20 04 c0  d2
0de1 : c1 c2 c3 c4 c5 c6 a4 20  f2
0de9 : 0a 50 51 52 53 54 a4 20  65
0df1 : 09 fa fb fc fd fe ff c7  7d
```

```
0df9 : a4 20 00 a4 20 00 a4 20  17
0e01 : 00 a4 20 bd c7 a4 20 04  3d
0e09 : d0 d1 d2 d3 d2 d5 d6 a4  71
0e11 : 20 05 7a 20 20 a4 6b 09  3d
0e19 : a4 87 04 a4 6c 09 c7 a4  8e
0e21 : 20 00 a4 20 00 a4 20 00  14
0e29 : a4 20 bd c7 a4 20 04 e0  63
0e31 : e1 e2 e3 e4 e5 e6 20 20  6f
0e39 : 20 a0 a1 a2 a1 a2 a3 ba  99
0e41 : bb ba bb bb ba bd 6c a4  54
0e49 : 91 04 6c 99 9a 96 99 9a  25
0e51 : 98 99 98 c7 a4 20 00 a4  69
0e59 : 20 00 a4 20 00 a4 20 bd  c8
0e61 : c7 a4 20 05 f1 f2 f3 f4  93
0e69 : f5 20 20 a0 a1 a3 b7 ba  16
0e71 : ba bb ba be ce cc ce cc  b7
0e79 : cd be 93 94 98 97 9a 95  f9
0e81 : 9a 99 9b 9c 9d 9e a5 ab  1f
0e89 : c7 a4 20 00 a4 20 00 a4  3f
0e91 : 20 00 a4 20 bd c7 90 90  5c
0e99 : 90 b7 a4 6b 0b d7 ce cd  e2
0ea1 : cc cd ce de dc 20 20 20  73
0ea9 : de 20 20 a5 a6 a7 a8 ab  f6
0eb1 : ac a4 20 06 c7 a4 20 00  9a
0eb9 : a4 20 00 a4 20 00 a4 20  d7
0ec1 : bd c7 20 20 20 c7 95 96  32
0ec9 : 97 98 99 9a 9b 99 9b 9d  96
0ed1 : 9e 8c a4 20 19 c7 a4 20  85
0ed9 : 00 a4 20 00 a4 20 00 a4  c8
0ee1 : 20 bd c7 90 90 90 d7 94  fa
0ee9 : 9b 9d 9e a5 a6 ab ac a4  73
0ef1 : 20 05 76 77 78 a4 20 04  56
0ef9 : e7 a4 20 04 f6 f7 f8 20  0e
0f01 : da da da a4 20 05 c7 a4  26
0f09 : 20 00 a4 20 00 a4 20 00  fc
0f11 : a4 20 bd c7 a4 20 10 6b  90
0f19 : d7 85 85 6c 73 73 73 6c  1b
0f21 : 72 72 72 6c 85 85 85 6c  6a
0f29 : a4 72 04 b7 92 92 92 c7  96
0f31 : a4 20 00 a4 20 00 a4 20  4f
0f39 : 00 a4 20 bd c7 a4 20 07  7b
0f41 : b0 b0 a4 20 06 6b 6b d7  90
0f49 : 45 46 a4 8c 10 20 c7 20  cd
0f51 : 20 20 c7 a4 20 00 a4 20  dd
0f59 : 00 a4 20 00 a4 20 bd c7  85
0f61 : 67 6a 6b a4 72 0a 67 6a  57
0f69 : 6b 6b d7 55 56 a4 20 11  58
0f71 : d7 92 92 92 c7 a4 20 00  aa
0f79 : a4 20 00 a4 20 00 a4 20  97
0f81 : bd c7 6c 67 a4 69 0c 6b  c6
0f89 : a0 a1 a2 a1 a2 a1 a3 a4  e6
0f91 : 20 04 67 a4 68 05 6c 90  a3
0f99 : 90 90 a4 20 05 c7 a4 20  00
0fa1 : 00 a4 20 00 a4 20 00 a4  90
0fa9 : 20 bd c7 20 20 45 46 a4  2c
0fb1 : 20 04 45 46 a4 20 04 45  d3
0fb9 : 46 a4 20 04 45 46 a4 20  33
0fc1 : 04 45 46 20 20 20 d7 d7  10
0fc9 : 90 90 90 a4 20 05 c7 a4  ed
0fd1 : 20 00 a4 20 00 a4 20 00  c4
0fd9 : a4 20 bd c7 20 20 55 56  fb
0fe1 : a4 20 04 55 56 a4 20 04  54
0fe9 : 55 56 a4 20 04 55 56 a4  24
0ff1 : 20 04 55 56 20 20 20 d7  67
0ff9 : d7 6c 6c a4 85 04 6c 6c  b9
1001 : c7 a4 20 00 a4 20 00 a4  b7
1009 : 20 00 a4 20 bd c7 92 92  e0
1011 : a0 a3 a4 85 23 a0 a3 c7  b2
1019 : a4 20 00 a4 20 00 a4 20  37
1021 : 00 a4 20 bd c7 a4 20 0c  6d
1029 : a5 a6 a7 a8 a7 a8 a7 ac  d8
1031 : 70 71 20 20 70 71 20 20  b9
1039 : 20 80 81 20 20 80 81 a4  53
1041 : 20 06 c7 a4 20 00 a4 20  c0
1049 : 00 a4 20 00 a4 20 bd c7  75
1051 : 92 92 a4 20 04 49 4a 4b  a4
1059 : 4c a4 20 0a 70 71 20 20  94
1061 : 70 71 20 20 20 80 81 20  e2
1069 : 20 80 81 a4 20 06 c7 a4  59
1071 : 20 00 a4 20 f2 24 a4 20  e2
1079 : 00 a4 20 ca c7 a4 20 04  57
1081 : 57 58 59 5a 5c 5c 5d 5e  81
1089 : a4 20 08 70 71 f0 f0 70  91
1091 : 71 20 20 20 80 81 20 20  f3
1099 : 80 81 20 20 88 89 8a 8b  fc
10a1 : c7 a4 20 00 a4 20 00 a4  57
10a9 : 20 00 a4 20 bd d7 a0 82  19
10b1 : 83 84 a2 a1 a2 a1 a2 a1  58
10b9 : a2 a1 a3 87 86 87 86 87  d3
10c1 : a0 a3 a0 a3 a0 a3 a0 a3  c0
10c9 : a4 6c 0c a0 85 a3 d7 a4  d9
10d1 : 20 23 a1 96 96 53 59 4d  c2
10d9 : 20 54 41 42 4c 45 20 4f  ca
10e1 : 56 45 52 46 4c 4f d7 49  68
10e9 : 45 45 c5 44 49 53 cb 20  69
10f1 : c8 9e a2 fa 9a a5 91 c9  c1
10f9 : 7f d0 07 20 db 9d 38 4c  ca
1101 : 34 a8 20 f8 a8 a0 ff 84  49
1109 : 3e 84 4e c8 84 3f b1 7a  34
1111 : d0 a4 08 06 0c 0c 08 08  48
1119 : a4 0f 04 08 08 08 a4 0b  b0
1121 : 04 08 0c 0c a4 0f 08 08  a1
1129 : 0d 0d 0d 0e 0b 0b a4 0e  7a
1131 : 06 0d 0e 0e 0e a4 0c 04  41
1139 : 0b 0b 0d 0d 0d a4 08 05  cf
1141 : 0f 0f 0e 0e a4 0a 04 0e  e4
1149 : 0e a4 0f 04 09 09 0f 0f  21
1151 : 0d 0c 0e a4 0f 0c 08 a4  37
1159 : 0a 04 08 a4 0f 07 0e 0f  7b
1161 : 0f 08 0b a4 0d 05 08 0c  fd
1169 : 08 a4 0a 06 08 a4 0d 07  ef
1171 : 0c 08 a4 0a 06 08 a4 0d  39
1179 : 07 0c 08 0e 0f 0f 0e 0e  08
1181 : 0e 08 a4 0d 07 0b a4 08  ca
1189 : 08 0b a4 0d 08 a4 0c 06  c3
1191 : 2c 36 00 36 93 20 a8 a8  cd
1199 : 20 a0 36 70 02 02 0a 20  3e
11a1 : 02 02 00 02 aa aa a0 a8  b9
11a9 : 8a 00 82 80 00 80 36 03  c7
11b1 : 80 36 0a aa 2f a4 0b 06  75
11b9 : aa f8 a4 e0 06 36 14 02  8b
11c1 : 0b 2f bf 02 0b 2e be fb  a9
11c9 : f9 e7 ed 80 a0 78 de 77  79
11d1 : dd 77 dd 36 04 80 e0 78  61
11d9 : de 36 18 aa 2a 36 06 aa  ef
11e1 : ff ab 02 02 02 0b 2f aa  01
11e9 : ff fe f8 f8 f8 fe 5f aa  1f
11f1 : fa a0 36 04 80 a0 80 36  c5
11f9 : 06 a4 0b 06 2f aa a4 e0  72
1201 : 06 f8 aa 36 04 02 0b 2f  d0
1209 : bf 02 0b 2f bf a4 ff 06  9f
1211 : fe fe fb f9 e7 ed b7 9d  d4
1219 : 77 dd 77 dd 77 dd 77 dd  18
1221 : 77 dd 77 dd 77 dd 77 dd  20
1229 : 77 dd 77 dd 77 de 80 e0  5b
1231 : 78 de 77 dd 77 dd 36 04  f9
1239 : 80 e0 78 de 36 08 9f be  c3
1241 : 2f 27 0a 36 03 de fb bf  e4
1249 : be 2e 08 00 00 fd fb 9e  3e
1251 : 2e 08 36 03 80 80 36 0b  6c
1259 : aa bf bf 36 05 a8 fe ff  2b
1261 : 36 07 80 aa 8c b3 8c b3  90
1269 : 8c b3 aa aa cc 33 cc 33  cf
1271 : cc 33 aa aa ce 32 ce 32  f5
1279 : ce 32 aa aa 8e b2 [illegible] b2  7e
1281 : 8e b2 aa aa 92 86 92 86  1d
1289 : 92 86 aa aa a4 be 06 aa  0c
1291 : ad 0a 36 06 65 d7 a6 08  51
1299 : 36 04 66 68 80 36 05 aa  9b
12a1 : a4 bf 06 aa aa a4 fe 06  d4
12a9 : aa a0 fa a4 ff 06 a0 fa  9f
12b1 : a4 ff 06 bf aa a4 02 05  b0
12b9 : 0a fe a8 a4 80 05 a0 00  b4
12c1 : 00 2a bc 8e bf 23 9a 00  57
12c9 : 00 0a a3 eb fe 5c 6b 00  b5
12d1 : 00 80 e0 38 f8 e0 a8 36  f6
12d9 : 0b 20 a8 20 20 20 36 17  2d
12e1 : 02 36 05 2a bf ff 36 05  64
12e9 : aa fe fe 36 11 a8 fe aa  40
12f1 : ff ff aa aa 00 02 8b aa  83
12f9 : fe fe aa aa 2a be fe aa  60
1301 : ff ff aa aa aa ff ff aa  00
1309 : ff ff aa aa 00 02 0b af  a3
1311 : ff ff aa aa 00 80 e8 fe  b5
1319 : ff ff aa aa aa bf ba bb  23
1321 : ba bb ba bf aa ff ff ff  0a
1329 : ee fb ee ff aa ff ef fe  39
1331 : ef ef ef ff aa fe be ae  0f
1339 : be be be fe 36 08 be 28  d5
1341 : 36 06 bf 2a a4 02 05 0a  32
1349 : fe aa a4 80 05 a0 aa ff  d5
1351 : aa ff aa ff aa ff aa aa  50
1359 : ff ff aa aa 55 55 36 08  41
1361 : 99 25 25 09 02 02 00 00  28
1369 : 51 55 5d 17 55 59 a5 0a  6a
1371 : 55 9d 55 59 55 57 1d 55  45
1379 : 56 15 5d 77 55 51 55 55  80
1381 : 57 5d 55 54 65 95 a4 55  a7
1389 : 04 57 71 d5 55 45 95 65  f0
1391 : 56 15 7d 57 d5 59 55 15  64
1399 : 55 5d 54 59 45 55 55 55  dc
13a1 : 69 56 d5 75 d5 56 a8 55  b7
13a9 : 55 75 5d 15 56 a8 00 55  08
13b1 : dd 65 5a 60 80 00 00 16  18
13b9 : 58 a0 36 0d aa bf bf aa  8e
13c1 : be be aa aa aa fe fe aa  d2
13c9 : ff ff aa aa aa ff ff aa  c8
13d1 : fe fe aa aa aa fe fe aa  42
13d9 : fe fe aa aa 36 08 9d 2b  c7
13e1 : 02 36 05 9f 75 a9 0a 36  6d
13e9 : 04 75 d6 58 60 80 36 03  51
13f1 : 55 5d 87 29 02 36 03 55  84
13f9 : f6 59 62 80 36 03 d4 7d  0e
1401 : 5a a0 36 04 5c 75 1a a0  d5
1409 : 36 04 6a 80 36 0a 20 28  71
1411 : aa aa 36 04 08 88 aa aa  e3
1419 : 36 09 28 a4 be 05 28 36  94
1421 : 06 02 0b 36 04 0a af ff  01
1429 : fe 36 03 0a af ff f3 ac  69
1431 : 36 03 aa ff bc fb e6 36  47
1439 : 03 80 e8 fe bf 7f 36 06  73
1441 : 80 80 aa b6 9e b6 9e b6  0a
1449 : 9e b6 36 08 bf bf 9f 2e  a6
1451 : 26 27 2f 2f 7f ff be ff  af
1459 : f7 ff eb 8b ff fd fb bb  13
1461 : bf bf fe de ff ff ef ef  3b
1469 : ed d9 db fb ff bd b7 ef  66
1471 : ef f7 f7 bd f6 96 9e be  2d
1479 : b8 b8 e8 e8 36 0c a4 02  3f
1481 : 04 2f 2f b3 bd ff fb fb  23
1489 : ee f9 63 ed ef ef ef 3a  bd
1491 : fe df 74 dd 55 d5 f6 fd  33
1499 : 97 bd f7 fd d3 f5 77 bd  13
14a1 : 66 b7 6f a5 e5 e7 6d bf  46
14a9 : 69 e0 e0 20 20 e0 f8 f8  9d
14b1 : f8 9e b6 9e b6 9e b6 9e  f2
14b9 : b6 36 08 2b 2f 2f 26 26  43
14c1 : 0a 0a 0a 82 82 82 36 05  c2
14c9 : d8 78 78 b8 a0 a0 20 20  e2
14d1 : bb bf bf bf 2e 2e 2e 0a  75
14d9 : bf af ae 88 08 36 03 68  3c
14e1 : e8 d8 98 a0 a0 20 20 36  67
14e9 : 08 02 0b 08 a4 0b 04 08  79
14f1 : fd df 79 de f9 bb bb fa  7a
14f9 : dc 7f 9f 7b 1e 4a fe 7c  15
1501 : dd bf d5 d7 dd f6 fd fe  ba
1509 : a5 e5 e7 6d bf 6b f5 f1  5b
1511 : af 9f 6d 1b 69 f9 79 db  53
1519 : 78 d8 78 f8 78 78 f8 78  5b
1521 : 9e b6 9e b6 9e b6 9e aa  08
1529 : 36 08 0a a4 08 04 36 03  fa
1531 : a4 08 04 28 28 2e b2 00  9e
1539 : 20 20 20 28 28 b8 be a4  03
1541 : 0a 04 a4 08 04 36 08 a4  d3
1549 : 20 04 36 0c 0b a4 02 04  60
1551 : 36 03 d9 7e ed fb f7 bd  69
1559 : 2f 0a df 77 95 75 dd ef  d0
1561 : fb 95 ae fe 77 dd b5 df  af
1569 : 3f d7 cf fd bb eb ad cb  b1
1571 : 3f d5 6f 9a a3 ff df 7e  81
1579 : c8 a0 e0 60 e0 80 80 36  56
1581 : 03 28 be aa b6 9e b6 9e  16
1589 : 28 36 11 80 82 22 22 28  33
1591 : 20 82 00 88 02 02 82 82  43
1599 : 88 08 00 80 22 20 a0 20  1b
15a1 : 20 82 00 a0 0a 88 88 a0  5f
15a9 : 80 02 00 a2 08 88 88 a0  a7
15b1 : 80 00 00 08 88 88 28 20  e0
15b9 : 20 80 36 05 08 2e be 36  a1
15c1 : 06 0a a5 29 02 02 09 09  c1
15c9 : 25 95 6a 55 56 59 6a 59  8b
15d1 : 55 aa 55 6a a0 80 80 80  2f
15d9 : 68 56 a5 36 06 a0 58 36  d0
15e1 : 03 0a 2f bd bf 23 36 03  61
15e9 : 02 ab ec bf fd 36 03 80  92
15f1 : e0 f8 78 e0 36 08 aa bf  55
15f9 : be be be bf be bf aa ff  52
1601 : ba fe be be be ff aa ff  58
1609 : ba fb fa fb fb ff aa ff  69
1611 : ba bb bb ba bb ff aa ff  55
1619 : fa be be fe be ff aa fe  96
1621 : be a4 fe 05 0e 09 00 4c  53
1629 : 6e c2 78 a9 e9 8d 14 03  ad
1631 : a9 c0 8d 15 03 a9 c0 8d  dc
1639 : e1 02 a9 fb 8d e0 02 a9  40
1641 : 2a 8d 12 d0 ad 11 d0 29  c9
1649 : 7f 8d 11 d0 a9 81 8d 1a  fe
1651 : d0 a2 28 a9 a0 9d 6f 07  74
1659 : a9 00 9d 6f db ca d0 f3  97
1661 : 20 c4 c2 a9 d0 8d 54 03  9a
1669 : ad 6b 03 f0 1f ad 68 03  b2
1671 : 8d 52 03 ad 69 03 8d 53  29
1679 : 03 ad 6a 03 8d 01 d0 ad  cd
1681 : 6c 03 85 fe ad 6d 03 85  0d
1689 : ff 4c 8b c0 a9 be 8d 52  15
1691 : 03 8d 68 03 a9 43 8d 53  67
1699 : 03 8d 69 03 a9 4d 8d 01  5b
16a1 : d0 8d 6a 03 a9 03 85 fe  fa
16a9 : 8d 6c 03 a9 05 85 ff 8d  fa
16b1 : 6d 03 ad 01 9f 8d 22 d0  bc
16b9 : a9 09 8d 05 d4 20 68 c2  60
16c1 : a9 01 8d 15 d0 8d 1c d0  7c
16c9 : a9 b0 8d 00 d0 ad 02 9f  ef
16d1 : 8d 23 d0 a9 00 8d 5a 03  35
16d9 : 8d 5c 03 8d 5b 03 8d 5e  c8
```

Listing. »Robos Revenge«. Bitte mit dem MSE eingeben

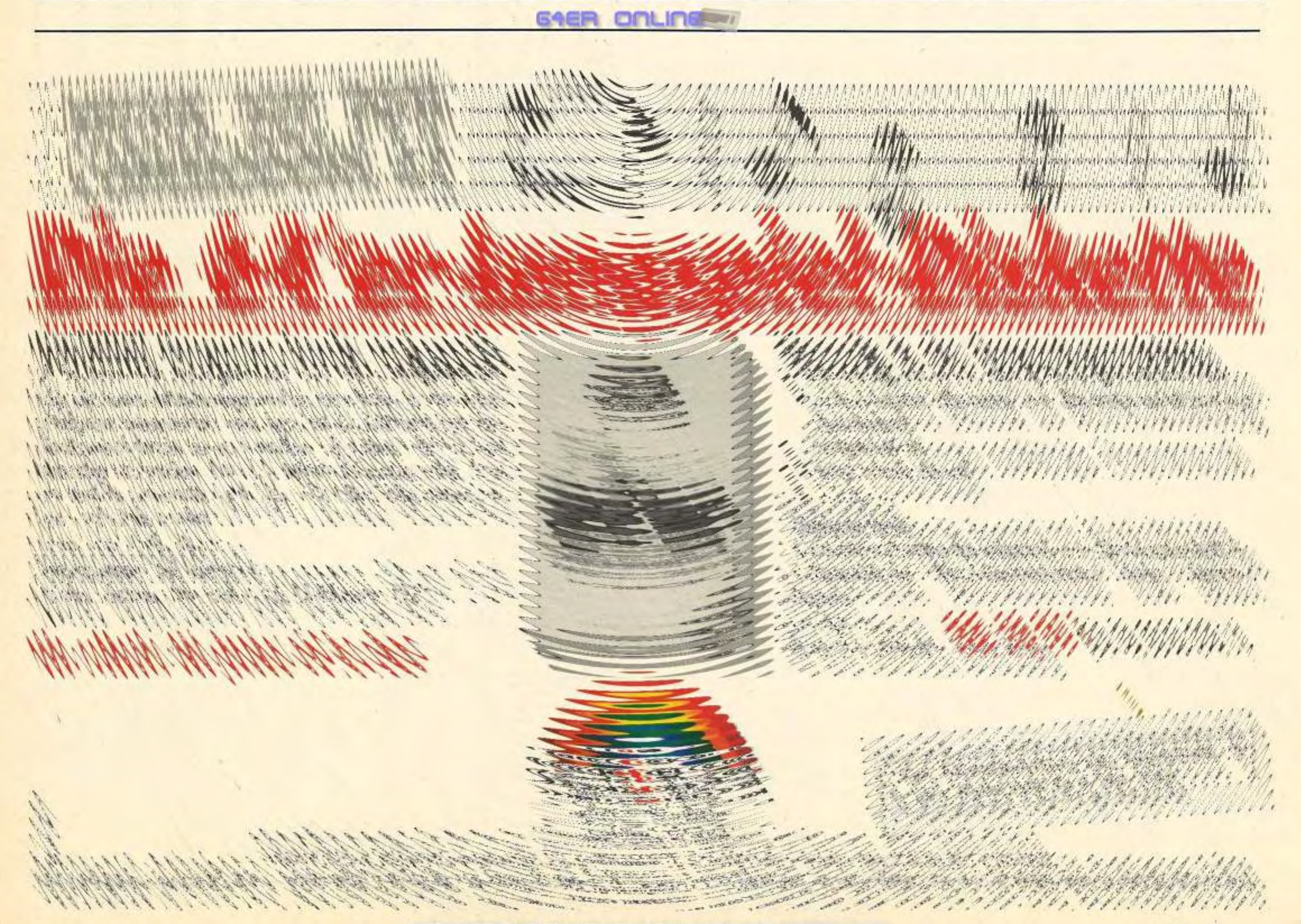


```
16e1 : 03 8d 56 03 8d 58 03 8d  64
16e9 : 59 03 8d 10 d0 8d 67 03  46
16f1 : 8d 04 d4 8d 06 d4 8d 00  a4
16f9 : d4 a9 18 8d 61 03 8d 03  c4
1701 : d4 a9 0f 8d 18 d4 a9 08  f4
1709 : 8d 01 d4 20 03 c4 58 60  c8
1711 : ad 19 d0 8d 19 d0 30 07  17
1719 : ad 0d dc 58 4c 31 ea 6c  62
1721 : e0 02 a9 0b 8d 20 d0 ad  47
1729 : 00 9f 8d 21 d0 ad 61 03  86
1731 : 8d 18 d0 ad 54 03 8d 16  74
1739 : d0 a9 2c 8d 12 d0 a9 00  e9
1741 : a9 ee 8d e0 02 a9 c1 8d  70
1749 : e1 02 20 2d c6 ad 60 03  3a
1751 : d0 1a ad 56 03 c9 01 d0  89
1759 : 63 a9 00 8d 67 03 ad 54  31
1761 : 03 c9 d1 f0 0a c9 d0 f0  ef
1769 : 2a ce 54 03 4c 7e ea a9  28
1771 : 00 8d 56 03 8d 65 03 a9  91
1779 : d0 8d 54 03 ad 5e 03 c9  f3
1781 : 02 d0 0d 20 e8 c4 ad 00  9e
1789 : dc c9 7f f0 03 20 f5 c4  da
1791 : 4c 7e ea ad 58 03 f0 03  f4
1799 : 4c 7e ea ad 52 03 c9 c0  7a
17a1 : d0 07 ad 53 03 c9 43 f0  38
17a9 : 10 ee 52 03 d0 03 ee 53  ad
17b1 : 03 a9 d7 8d 54 03 8d 57  73
17b9 : 03 4c 7e ea c9 02 d0 f9  c3
17c1 : a9 00 8d 67 03 ae 54 03  b8
17c9 : e0 d7 f0 13 e8 e0 d1 d0  b2
17d1 : 08 20 7f c4 ad 56 03 f0  dd
17d9 : 03 ee 54 03 4c 7e ea ad  89
17e1 : 58 03 f0 03 4c 7e ea ad  17
17e9 : 52 03 c9 29 d0 07 ad 53  f7
17f1 : 03 c9 40 f0 18 ce 52 03  4e
17f9 : ad 52 03 c9 ff d0 03 ce  f9
1801 : 53 03 a9 d0 8d 54 03 8d  fd
1809 : 65 03 8d 57 03 4c 7e ea  a0
1811 : a9 00 8d 1f d0 a9 e2 8d  03
1819 : 12 d0 a9 23 8d e0 02 a9  9d
1821 : c2 8d e1 02 58 ad 60 03  dd
1829 : f0 03 4c 7e ea ad 57 03  fd
1831 : f0 15 ad 56 03 c9 01 f0  46
1839 : 05 a9 00 8d 56 03 20 e1  86
1841 : c2 20 68 c2 4c 7e ea 4c  83
1849 : db c2 a9 00 8d 20 d0 a9  60
1851 : 06 8d 21 d0 a9 15 8d 18  2a
1859 : d0 a9 c8 8d 16 d0 a9 2a  c4
1861 : 8d 12 d0 a9 fb 8d e0 02  14
1869 : a9 c0 8d e1 02 ad 1f d0  be
1871 : 8d 66 03 ad 60 03 f0 0e  a6
1879 : ad 58 03 d0 06 20 00 c7  1e
1881 : 20 2d c6 4c 7e ea 20 0e  4f
1889 : c4 20 42 c4 4c 51 c2 ad  3c
1891 : 58 03 f0 01 60 a9 a4 36  19
1899 : 01 85 01 a9 00 8d 57 03  a2
18a1 : ad 52 03 8d 8b c2 ad 53  16
18a9 : 03 8d 8c c2 a9 16 8d 58  20
18b1 : 03 a2 28 bd 00 40 9d 00  40
18b9 : 04 a8 b9 00 96 9d 00 d8  88
18c1 : ca d0 f0 ad 8b c2 18 69  e7
18c9 : e8 8d 8b c2 ad 8c c2 69  d0
18d1 : 03 8d 8c c2 ad 8e c2 18  a1
18d9 : 69 28 90 06 ee 8f c2 ee  8f
18e1 : 96 c2 8d 8e c2 8d 95 c2  82
18e9 : ce 58 03 d0 c4 a9 d7 8d  d2
18f1 : 96 c2 a9 03 8d 8f c2 a9  67
18f9 : ff 8d 8e c2 8d 95 c2 a9  9f
1901 : 37 85 01 60 20 e1 c2 4c  fc
1909 : 7e ea ae 59 03 e8 ee 67  d5
1911 : 03 8e 59 03 e0 21 d0 19  9f
1919 : a2 00 8e 04 d4 ca 8e 64  86
1921 : 03 20 03 c4 a9 03 8d 01  79
1929 : d4 a2 41 8e 04 d4 ae 59  c5
1931 : 03 bd 3d c6 f0 69 8d 5d  86
1939 : 03 ad 59 03 c9 34 f0 07  da
1941 : c9 11 f0 03 4c 6c c3 a5  b2
1949 : ff 48 a5 fe 48 38 e9 28  f4
1951 : b0 02 c6 ff 85 fe ad 59  6e
1959 : 03 48 ad 56 03 48 20 1b  e0
1961 : c4 68 8d 56 03 68 8d 59  e4
1969 : 03 ad 5a 03 d0 1e 68 68  aa
1971 : ad 01 d0 38 e9 08 a2 20  84
1979 : 8e 59 03 a2 cb 8e 5d 03  76
1981 : 8e f8 07 8d 01 d0 20 39  88
1989 : c4 4c 6c c3 68 85 fe 68  86
1991 : 85 ff 20 39 c4 ad 5d 03  7a
1999 : 8d f8 07 20 a1 c3 60 ad  7d
19a1 : 67 03 c9 41 d0 1c ad 52  6e
19a9 : 03 8d 68 03 ad 53 03 8d  8a
19b1 : 69 03 ad 01 d0 8d 6a 03  50
19b9 : a5 fe 8d 6c 03 a5 ff 8d  47
19c1 : 6d 03 a2 00 8e 67 03 4c  21
19c9 : e8 c2 20 77 c4 20 7f c4  de
19d1 : ad 5f 03 c9 02 f0 2e c9  1c
19d9 : 01 f0 07 a9 04 2c 00 dc  a5
19e1 : d0 1c ad 5e 03 c9 01 f0  5b
19e9 : 0f a2 02 8e 56 03 a9 00  c0
19f1 : 8d 5f 03 8d 5e 03 f0 3e  df
19f9 : 20 e8 c4 4c f5 c4 a9 08  84
1a01 : 2c 00 dc d0 27 ad 5e 03  de
1a09 : c9 02 f0 0e a2 01 8e 56  ea
1a11 : 03 ca 8e 5f 03 8e 5e 03  2d
1a19 : f0 1c ad 54 03 c9 d0 d0  71
1a21 : 15 20 e8 c4 ad 65 03 d0  cd
1a29 : 0d 4c f5 c4 a9 04 8d b3  cb
1a31 : c3 a9 08 8d d6 c3 60 ad  e5
1a39 : 59 03 c9 20 f0 01 60 ad  7e
1a41 : 5a 03 d0 fa a0 00 a2 00  45
1a49 : bd 7f c6 f0 07 d1 fe f0  72
1a51 : 11 e8 d0 f4 a9 1f 8d 59  25
1a59 : 03 a9 01 8d 5a 03 8d 5b  ce
1a61 : 03 60 a9 00 8d 5a 03 8d  d2
1a69 : 5b 03 60 ad 5b 03 c9 01  0b
1a71 : f0 01 60 ee 01 d0 ee 5c  e3
1a79 : 03 a9 1f 8d 59 03 8d 67  7d
1a81 : 03 ad 5c 03 c9 08 f0 01  75
1a89 : 60 a9 00 8d 5c 03 8d 67  52
1a91 : 03 a5 fe 18 69 28 90 02  48
1a99 : e6 ff 85 fe 20 1b c4 60  6f
1aa1 : ad 58 03 d0 03 20 1d c5  86
1aa9 : a9 00 8d 5e 03 ad 5d 03  9b
1ab1 : c9 c4 b0 0d a9 4f 8d ca  8b
1ab9 : c4 a9 51 8d b4 c4 4c a5  46
1ac1 : c4 a9 27 8d ca c4 a9 29  a1
1ac9 : 8d b4 c4 4c a5 c4 ad 56  4f
1ad1 : 03 c9 01 f0 16 a5 ff 85  b1
1ad9 : fd a5 fe 38 e9 29 b0 02  1e
1ae1 : c6 fd 85 fc 20 09 c5 c9  9c
1ae9 : 01 f0 17 a5 ff 85 fd a5  4c
1af1 : fe 38 e9 27 b0 02 c6 fd  9d
1af9 : 85 fc 20 09 c5 c9 01 f0  b6
1b01 : 0a 60 8d 5e 03 a9 00 8d  03
1b09 : 56 03 60 a9 02 8d 5e 03  3a
1b11 : 60 ad b3 c3 ae d6 c3 8d  79
1b19 : d6 c3 8e b3 c3 60 a2 00  b5
1b21 : 8e 04 d4 ca 8e 64 03 a9  ab
1b29 : 04 8d 01 d4 a9 41 8d 04  b2
1b31 : d4 60 a2 00 a0 00 bd 94  08
1b39 : c6 f0 07 d1 fc f0 04 e8  ad
1b41 : d0 f4 60 a9 01 60 a5 ff  82
1b49 : 85 fd a5 fe 85 fc ad 59  c0
1b51 : 03 c9 14 90 04 c9 2f 90  bc
1b59 : 0b a5 fc 38 e9 28 b0 02  24
1b61 : c6 fd 85 fc a5 fc 38 e9  1e
1b69 : 28 b0 02 c6 fd 85 fc a9  96
1b71 : 90 8d 91 c6 a9 92 8d 92  8f
1b79 : c6 a0 00 b1 fe c9 90 d0  e8
1b81 : 12 ad 80 24 d0 58 a9 92  aa
1b89 : 8d 91 c6 a9 00 8d 92 c6  0a
1b91 : 4c b5 c5 c9 92 d0 12 ad  b6
1b99 : 90 24 d0 42 a9 90 8d 91  30
1ba1 : c6 a9 00 8d 92 c6 4c b5  ea
1ba9 : c5 c9 72 f0 12 c9 82 90  a8
1bb1 : 1d c9 85 b0 19 ad 59 03  95
1bb9 : c9 20 d0 27 4c d5 c5 ad  91
1bc1 : 59 03 c9 20 d0 1d a9 02  b3
1bc9 : 8d 5f 03 4c b5 c5 c9 73  e8
1bd1 : d0 11 ad 59 03 c9 20 d0  61
1bd9 : 05 a9 01 8d 5f 03 a9 00  5a
1be1 : 8d 67 03 b1 fc c9 70 d0  9a
1be9 : 08 ad 80 23 f0 40 4c d5  3a
1bf1 : c5 c9 71 f0 f4 c9 80 d0  56
1bf9 : 13 ad 00 24 f0 30 a9 02  a3
1c01 : 8d 62 03 8d 60 03 8d 67  55
1c09 : 03 4c 05 c6 c9 81 f0 e9  8d
1c11 : c9 88 90 0f c9 8c b0 0b  fe
1c19 : a9 01 8d 62 03 8d 60 03  17
1c21 : 4c 05 c6 c9 86 f0 04 c9  6e
1c29 : 87 d0 08 4c d5 c5 a9 00  d6
1c31 : 8d 67 03 a2 00 bd 98 c6  65
1c39 : f0 07 d1 fc f0 04 e8 d0  35
1c41 : f4 60 ad 66 03 f0 0e a9  e1
1c49 : 02 8d 62 03 8d 60 03 8d  0e
1c51 : 67 03 ee 20 d0 60 ad 64  89
1c59 : 03 38 e9 02 8d 64 03 8d  56
1c61 : 02 d4 8d 00 d4 60 a4 c0  95
1c69 : 04 a4 c1 05 a4 c2 05 a4  8e
1c71 : c3 04 c4 c4 c4 c5 c5 c5  1d
1c79 : c6 c6 c7 c7 c8 c8 c9 ca  1d
1c81 : cb cc cd cc cb ca c9 c8  8b
1c89 : c8 c7 c7 c6 c6 c5 c5 c5  3d
1c91 : c4 c4 c4 a4 c3 04 a4 c2  f1
1c99 : 05 a4 c1 05 00 a0 a1 a2  d2
1ca1 : a3 72 73 85 6c 82 83 84  fd
1ca9 : b7 67 68 69 6a 6b 91 90  c4
1cb1 : 92 00 c7 b7 d7 00 b0 76  59
1cb9 : 77 78 b0 ad ae b9 ba bb  69
1cc1 : bc bd be c9 ca cb cc cd  1e
1cc9 : ce d9 da db dc de f6 f7  46
1cd1 : f8 e7 f0 00 ad 6f 03 f0  3d
1cd9 : 01 60 20 26 c7 ad 76 03  a1
1ce1 : c9 03 b0 04 ee 76 03 60  48
1ce9 : a9 00 8d 76 03 20 90 c7  c7
1cf1 : 20 5e c7 20 a6 c7 20 1d  9a
1cf9 : c8 60 ad 70 03 c9 03 b0  57
1d01 : 04 ee 70 03 60 a2 00 8e  31
1d09 : 70 03 20 41 c7 20 41 c7  3d
1d11 : a2 01 20 41 c7 18 3e 98  cb
1d19 : 23 90 08 bd 98 23 09 01  07
1d21 : 9d 98 23 18 7e 90 23 90  f0
1d29 : 08 bd 90 23 09 80 9d 90  c5
1d31 : 23 60 20 61 c7 a2 05 18  8e
1d39 : 3e 30 24 90 21 20 77 c7  2b
1d41 : bd 38 24 09 01 9d 38 24  6b
1d49 : 4c 86 c7 18 3e 38 24 90  25
1d51 : 08 bd 30 24 09 01 9d 30  38
1d59 : 24 60 ca d0 da 60 20 77  9a
1d61 : c7 4c 86 c7 ad 88 24 a8  ea
1d69 : a2 00 bd 89 24 9d 88 24  45
1d71 : e8 e0 07 d0 f5 98 8d 8f  1f
1d79 : 24 60 ad 74 03 c9 02 f0  30
1d81 : 04 ee 74 03 60 a9 00 8d  e8
1d89 : 74 03 ad 72 03 d0 31 ae  11
1d91 : 71 03 e0 ff d0 13 a2 00  ec
1d99 : 8e 73 03 a2 07 8e 71 03  a7
1da1 : 8e 72 03 a2 fe 8e 74 03  ba
1da9 : 60 bd 80 24 a8 a9 00 9d  a0
1db1 : 80 24 ae 73 03 98 9d 90  ea
1db9 : 24 ce 71 03 ee 73 03 60  58
1dc1 : ae 71 03 e0 ff d0 13 a2  1d
1dc9 : 00 8e 72 03 8e 73 03 a2  e3
1dd1 : 07 8e 71 03 a2 fe 8e 74  21
1dd9 : 03 60 bd 90 24 a8 a9 00  bc
1de1 : 9d 90 24 ae 73 03 98 9d  92
1de9 : 80 24 ce 71 03 ee 73 03  d9
1df1 : 60 ad 75 03 c9 19 b0 04  16
1df9 : ee 75 03 60 a9 00 8d 75  2a
1e01 : 03 aa bd 00 24 a8 bd 80  48
1e09 : 23 9d 00 24 98 9d 80 23  3e
1e11 : e8 e0 10 d0 ed 60 4c 0c  b3
1e19 : cc a9 0e 8d 00 9f a9 00  93
1e21 : 8d 02 9f a9 09 8d 01 9f  0d
1e29 : a9 01 8d 6f 03 20 0c cc  9f
1e31 : a9 00 8d 6f 03 20 03 c0  ea
1e39 : a9 01 8d 60 03 58 a9 15  96
1e41 : 8d 61 03 a2 00 8e 15 d0  fe
1e49 : e8 8e 0e dc a9 d8 8d 54  d8
1e51 : 03 ad 00 9f 8d 6e 03 a9  ca
1e59 : 0e 8d 00 9f a9 9d 8d a1  23
1e61 : c8 a9 c9 8d a2 c8 ad 9d  84
1e69 : c9 f0 0e 20 d2 ff ee a1  5e
1e71 : c8 d0 f3 ee a2 c8 4c a0  5f
1e79 : c8 a5 c5 c9 38 f0 1c c9  cd
1e81 : 3b d0 03 4c 70 c9 c9 10  0b
1e89 : d0 03 4c e2 fc c9 08 d0  2a
1e91 : 03 4c 42 cc c9 0b d0 e1  e0
1e99 : 4c 4b cc ad 6e 03 8d 00  a9
1ea1 : 9f a9 00 8d 6b 03 8d 60  8c
1ea9 : 03 8d 62 03 8d 0e dc 20  69
1eb1 : 03 c0 58 a9 10 2c 00 dc  7c
1eb9 : d0 03 4c 62 c8 ad 62 03  f4
1ec1 : f0 f1 c9 02 f0 28 a9 04  5b
1ec9 : 8d 10 c9 a9 cb 8d 11 c9  07
1ed1 : a9 15 8d 61 03 ad 04 cb  da
1ed9 : f0 0e 20 d2 ff ee 10 c9  7e
1ee1 : d0 f3 ee 11 c9 4c 0f c9  57
1ee9 : 8d 15 d0 4c 89 c9 ce c3  68
1ef1 : 07 ce e4 07 ad c3 07 c9  22
1ef9 : 30 f0 2a a2 00 bd f6 cb  e2
1f01 : f0 06 20 d2 ff e8 d0 f5  cd
1f09 : ad 00 dc c9 7d d0 f9 a2  b2
1f11 : 00 bd fc cb f0 06 20 d2  0e
1f19 : ff e8 d0 f5 a2 01 8e 6b  c2
1f21 : 03 ca 8a f0 81 a2 00 bd  f3
1f29 : c2 cb f0 07 20 d2 ff e8  58
1f31 : 4c 61 c9 4c 89 c9 ad 6e  a4
1f39 : 03 8d 00 9f a9 93 20 d2  54
1f41 : ff 20 03 c0 a9 18 8d 61  7e
1f49 : 03 20 a3 cc 4c 62 c8 a0  1b
1f51 : 00 a2 00 e8 d0 fd c8 d0  81
1f59 : f8 ad 00 dc c9 6f d0 f9  13
1f61 : 4c 62 c8 93 1c 11 a4 1d  9a
1f69 : 0c 52 4f 42 4f 27 53 20  76
1f71 : 52 45 56 45 4e 47 45 0d  f2
1f79 : 0d 9c a4 1d 0d 50 52 4f  dc
1f81 : 47 52 41 4d 4d 49 45 52  c4
1f89 : 54 0d 11 a4 1d 11 56 4f  8f
1f91 : 4e 0d 11 a4 1d 04 1f 52  52
1f99 : 4f 42 45 52 54 20 42 52  99
1fa1 : 41 4e 44 4c 20 55 4e 44  12
1fa9 : 20 54 48 4f 4d 41 53 20  5c
1fb1 : 53 43 48 4d 49 44 54 0d  84
1fb9 : 11 90 31 20 3d 20 53 50  25
1fc1 : 49 45 4c 45 4e 20 20 20  bf
1fc9 : 3d 20 45 44 49 54 49 45  d7
1fd1 : 52 45 4e 0d 33 20 3d 20  65
1fd9 : 4c 41 44 45 4e 0d 34 20  de
1fe1 : 3d 20 53 41 56 45 4e 0d  0e
1fe9 : 35 20 3d 20 45 4e 44 45  e4
```

Listing. »Robos Revenge« (Fortsetzung)

```
1ff1 : 0d a4 11 04 a4 1d 0c 21  bb
1ff9 : 20 56 49 45 4c 20 53 50  f3
2001 : 41 53 53 20 21 0d 11 11  a6
2009 : 11 12 a4 20 28 92 9c c3  62
2011 : 9e 52 4f 42 4f 53 9c c0  78
2019 : 1d 1d 1d 9f 2a 20 52 4f  8c
2021 : 42 4f 27 53 20 52 45 56  95
2029 : 45 4e 47 45 20 2a a4 1d  30
2031 : 05 9c c0 9e 52 4f 42 4f  cf
2039 : 53 9c c3 1d 1d 1d 9f 33  0e
2041 : a4 1d 20 33 a4 9d 15 00  6e
2049 : 93 90 11 a4 1d 0b 21 21  ee
2051 : 21 20 47 52 41 54 55 4c  43
2059 : 49 45 52 45 20 21 21 21  54
2061 : 0d 11 a4 1d 0a 53 49 45  af
2069 : 20 48 41 42 45 4e 20 52  32
2071 : 4f 42 4f 45 47 47 20 49  20
2079 : 4e 0d 11 a4 1d 09 53 49  21
2081 : 43 48 45 52 48 45 49 54  00
2089 : 20 20 20 47 45 42 52 41  dd
2091 : 43 48 54 2e 0d 11 a4 1d  f9
2099 : 12 41 42 45 52 0d 11 1d  91
20a1 : 1d 57 45 52 44 45 4e 20  ed
20a9 : 53 49 45 20 41 55 43 48  53
20b1 : 20 53 43 48 57 49 45 52  ce
20b9 : 49 47 45 52 45 20 4c 45  53
20c1 : 56 45 4c 53 0d 11 11 a4  1e
20c9 : 1d 0e 3f 20 4d 45 49 53  8c
20d1 : 54 45 52 4e 20 3f 00 13  48
20d9 : a4 11 17 a4 1d 0b 05 a4  e8
20e1 : 20 04 47 41 4d 45 20 4f  1b
20e9 : 56 45 52 20 20 00 43 52  2e
20f1 : 41 53 48 00 a4 9d 05 a4  83
20f9 : 20 05 a4 9d 05 00 a9 07  7e
2101 : 8d 71 03 a9 00 8d 70 03  71
2109 : 8d 72 03 8d 73 03 8d 74  b0
2111 : 03 8d 75 03 aa a0 08 a9  bc
2119 : 97 8d 31 cc a9 20 8d 34  97
2121 : cc bd 00 97 9d 00 20 e8  eb
2129 : d0 f7 ee 31 cc ee 34 cc  85
2131 : 88 d0 ee 60 20 54 cc 20  01
2139 : a7 f4 4c 90 cc 20 54 cc  38
2141 : 20 d8 ff 4c 90 cc a9 00  6d
2149 : 85 c6 c5 c6 f0 fc ad 77  18
2151 : 02 8d 98 cc c9 30 30 28  09
2159 : c9 3a 10 24 a2 08 a0 01  b7
2161 : 20 ba ff a9 01 a2 98 a0  dc
2169 : cc 20 bd ff a9 00 8d 11  a8
2171 : d0 85 93 85 02 a9 40 85  13
2179 : 03 a2 03 a0 9f a9 02 60  b2
2181 : 68 68 a9 1b 8d 11 d0 4c  28
2189 : 62 c8 00 4c a9 cc 4c 4d  a6
2191 : ce a9 d8 8d 54 03 a9 17  4e
2199 : 8d 00 d0 a9 31 8d 01 d0  b5
21a1 : a9 0f 8d f8 07 a2 01 8e  fb
21a9 : 15 d0 ca 8e 1c d0 8e 10  4d
21b1 : d0 a9 04 85 03 a9 00 85  90
21b9 : 02 a2 ac 8e 52 03 e8 86  f7
21c1 : 04 a9 43 85 05 8d 53 03  2b
21c9 : 20 d0 ce a9 00 85 c6 a8  d3
21d1 : ee 27 d0 c5 c6 f0 f9 ad  77
21d9 : 77 02 d9 d8 ce f0 0a c8  11
21e1 : c8 c8 c0 24 d0 f4 4c e3  70
21e9 : cc b9 d9 ce 85 f7 b9 da  97
21f1 : ce 85 f8 6c f7 00 ad 00  84
21f9 : d0 18 69 08 b0 11 c9 57  9a
2201 : d0 07 ad 10 d0 d0 1f a9  26
2209 : 57 8d 00 d0 4c 2f cd a9  0a
2211 : 07 8d 00 d0 ee 10 d0 e6  79
2219 : 02 d0 02 e6 03 e6 04 d0  fa
2221 : 02 e6 05 4c e3 cc ad 52  61
2229 : 03 c9 ac d0 0a ad 53 03  b7
2231 : c9 43 d0 03 4c e3 cc ee  25
2239 : 52 03 d0 03 ee 53 03 20  77
2241 : d0 ce 4c 35 cd ad 00 d0  1e
2249 : 38 e9 08 90 11 c9 0f d0  c7
2251 : 07 ad 10 d0 f0 23 a9 0f  3a
2259 : 8d 00 d0 4c 7e cd a9 ff  a1
2261 : 8d 00 d0 ce 10 d0 a5 02  1e
2269 : d0 02 c6 03 c6 02 a5 04  67
2271 : d0 02 c6 05 c6 04 4c e3  1a
2279 : cc ad 52 03 c9 00 d0 0a  05
2281 : ad 53 03 c9 40 d0 03 4c  01
2289 : e3 cc ad 52 03 d0 03 ce  e8
2291 : 53 03 ce 52 03 20 d0 ce  76
2299 : 4c 86 cd ad 01 d0 18 69  1b
22a1 : 08 c9 e1 f0 1b 8d 01 d0  e8
22a9 : a5 02 18 69 28 85 02 90  5a
22b1 : 02 e6 03 a5 04 18 69 e8  14
22b9 : 85 04 a5 05 69 03 85 05  19
22c1 : 4c e3 cc ad 01 d0 38 e9  33
22c9 : 08 c9 29 f0 1b 8d 01 d0  e2
22d1 : a5 02 38 e9 28 85 02 b0  db
22d9 : 02 c6 03 a5 04 38 e9 e8  2f
22e1 : 85 04 a5 05 e9 03 85 05  49
22e9 : 4c e3 cc a0 00 b1 02 8d  1f
22f1 : fc ce 4c e0 cc a0 00 ad  b1
22f9 : fc ce 91 02 91 04 aa a5  30
2301 : 03 18 69 d4 85 fa a5 02  d0
2309 : 85 f9 bd 00 96 91 f9 4c  71
2311 : e3 cc a2 00 8e 15 d0 e8  aa
2319 : 8e 27 d0 60 20 3a ce 4c  23
2321 : e3 cc a9 ff 8d 00 cf a9  40
2329 : d4 85 f7 85 f9 a9 04 85  76
2331 : f8 a9 d8 85 fa a2 30 a0  ab
2339 : 00 8a 91 f7 bd 00 96 91  3b
2341 : f9 e8 f0 20 c8 c0 10 d0  63
2349 : f0 a0 00 a5 f7 18 69 28  74
2351 : 90 02 e6 f8 85 f7 a5 f9  5e
2359 : 18 69 28 90 02 e6 fa 85  90
2361 : f9 4c 51 ce 60 a0 00 a2  ff
2369 : 56 84 f7 a9 40 85 f8 a9  9c
2371 : 20 91 f7 c8 d0 fb e6 f8  eb
2379 : ca d0 f6 4c a9 cc ad 00  aa
2381 : d0 8d fd ce ad 01 d0 8d  b2
2389 : fe ce ad 10 d0 8d ff ce  73
2391 : 20 00 10 ad fd ce 8d 00  f8
2399 : d0 ad fe ce 8d 01 d0 a9  51
23a1 : 01 8d 15 d0 ad ff ce 8d  f9
23a9 : 10 d0 20 00 c0 ad 00 cf  43
23b1 : f0 03 4c 34 ce 4c e3 cc  35
23b9 : a9 00 8d 00 cf 4c 00 c0  a7
23c1 : 1d 0e cd 9d 5d cd 91 db  4f
23c9 : cd 11 b3 cd 5a 03 ce da  74
23d1 : 03 ce 58 0d ce d8 0d ce  79
23d9 : 03 2a ce 20 34 ce 93 7d  ac
23e1 : ce 5f 96 ce 36 05 ff c0  eb
23e9 : 00 80 40 00 80 40 00 80  45
23f1 : 40 00 80 40 00 80 40 00  5f
23f9 : 80 40 00 80 40 00 80 40  30
2401 : 00 ff c0 36 23 00 02 b3  99
```

Listing. »Robos Revenge« (Schluß)



Vorsicht vor den Minen!

In diesem Spiel sind Sie ein gefährlich lebender Abenteurer. Nur mit einem ungenauen Minendetektor ausgerüstet, müssen Sie ein von hochexplosiven Minen übersätes Feld überqueren. Eine High-Score-Liste auf Diskette gehört natürlich auch dazu. Falls Sie kniffelige und ausgefallene Strategiespiele lieben, ist »Minefield« genau das richtige.

Der Name »Minefield«, wie dieses Spiel heißt, dürfte vielen bekannt sein, die schon einmal an einem Großrechner mit Unix-Betriebssystem gearbeitet haben. Minefield gehört nämlich mit zum Lieferumfang des Betriebssystems. Wer es einmal gespielt hat, den läßt es so schnell nicht mehr los.

Zur Spielweise: Das Spielfeld besteht aus 30 mal 15 Feldern. Auf jedem Feld kann sich theoretisch eine (unsichtbare!) Mine befinden. Wenn Sie auf eine derselben treten, fliegen Sie in die Luft. Der Spieler beginnt in der linken, oberen Ecke und muß zum Ausgang ganz rechts unten kommen. Die Steuerung Ihrer Spielfigur wurde auf der Tastatur in Form eines Koordinatenkreuzes verwirklicht (Bild 1).

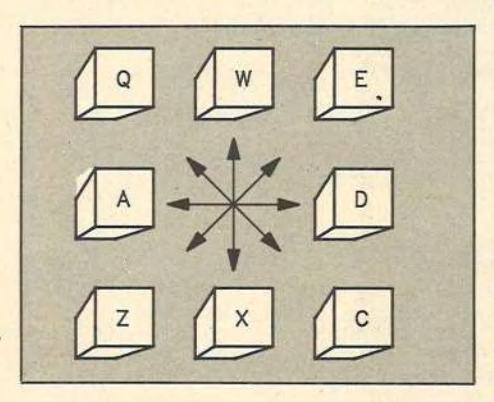


Bild 1. Mit diesen Tasten bewegen Sie Ihre Spielfigur durch das gefährliche Minenfeld

Sie können also nicht nur waagrecht und senkrecht ziehen, sondern auch diagonal. Auf dem Feld, auf dem sich der Spieler gerade befindet, wird invers (hell auf dunkel) eine Zahl angezeigt. Diese Zahl ist die Anzeige Ihres Minendetektors. Sie können sich das so vorstellen: Jede Mine im Umkreis von einem Feld sendet ein Signal, das sich aber nicht genau orten läßt. Der Detektor kann nur messen, wie viele Signale eintreffen, sprich, wie viele Minen im Umkreis von einem Feld vorhanden sind. Um dies besser zu verstehen, sollten Sie Bild 2 betrachten.

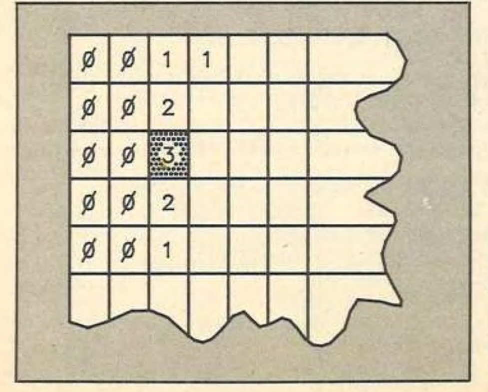


Bild 2. Eine typische Spielsituation. Auf welchen Feldern befinden sich denn nun die Minen?

Angenommen, Sie befinden sich gerade an der Position der 3. Das bedeutet, daß irgendwo um Sie herum drei Minen versteckt sind. In diesem Falle ist die Verteilung eindeutig, da sowieso nur noch drei Felder in direkter Nähe frei sind (rechts oben, rechts in der Mitte und rechts unten von Ihrer Position aus gesehen). Also liegen dort auch die drei Minen, deren Signale Sie gerade empfangen.

Spielstart

Wenn das Programm mit RUN gestartet wurde, erscheint als erstes ein kleines Auswahlmenü. Beim ersten Start müssen anfangs die High-Score-Dateien auf der Diskette eingerichtet werden. Drücken Sie dazu <C>. Nach ein paar Sekunden erscheint wieder das Titelbild.

Nun wollen wir spielen: Um mit der Zählweise etwas vertraut zu werden, ist in Minefield ein Trainermodus eingebaut. Dieser bewirkt, daß alle Minen auf dem Bildschirm als Sternchen (*) angezeigt werden. Sie können so um die Minen herumgehen und die Anzeige Ihres Minendetektors beobachten. Wenn Sie den Ausgang rechts unten erreicht haben, werden Sie allerdings nicht in die High-Score-Liste aufgenommen (es sollte ja auch nur zum Üben sein). Der Trainermodus wird mit der Taste <T> ein- und ausgeschaltet.

Die Schwierigkeitsgrade

Es gibt vier Schwierigkeitsgrade: 1 - easy (leicht), 2 - medium (mittel), 3 - hard (schwer) und 4 - suicide (Selbstmord). Abhängig von der gewählten Schwierigkeitsstufe wird vom Programm die Anzahl an Minen festgelegt. Um bei Stufe 4 durchzukommen, müssen Sie schon einige Zeit gespielt haben. Um das Spiel zu starten, stellen Sie eventuell den Trainermodus ein und drücken dann eine Taste von <1> bis <4>. Nach ein paar Sekunden, in denen der C64 die Minen versteckt, wird das Spielfeld dargestellt, und es kann losgehen.

Der Zeichenstift

Als Hilfe steht Ihnen ein Zeichenstift zur Markierung von potentiellen Minen auf dem Bildschirm zur Verfügung: Drücken Sie <←> (Pfeil nach links). Mit den normalen Steuertasten kann jetzt eine Markierung über den Bildschirm bewegt werden. Wenn Sie <S> drücken, wird an dieser Stelle eine Markierung gesetzt beziehungsweise gelöscht. Natürlich können Sie Markierungen nur auf Felder setzen, auf denen Sie noch nicht waren. Erneutes Drücken von <←> bringt Sie wieder in den normalen Spielmodus. Anhand dieser Markierungen läßt sich dann viel leichter ausrechnen, an welchen Stellen sich noch Minen befinden müssen.

Spielende

Wenn Sie den Ausgang rechts unten erreicht haben, bekommen Sie für jedes Feld, auf dem Sie waren, einen vom Schwierigkeitsgrad abhängigen Punkt. Auf Stufe 3 zum Beispiel gibt es für jedes überlaufene Feld drei Punkte. Zusätzlich gibt es für jede vollständig eingekreiste Mine einen Bonus. Das bedeutet, daß rings um eine Mine kein Feld ausgelassen sein darf, außer, es befindet sich dort wieder eine Mine (siehe Bild 3).

Die hier als * eingezeichneten Minen sehen Sie im normalen Spielablauf natürlich nicht. Beide Minen gelten als vollständig eingekreist. Bei der Punktezählung würden Sie also zweimal einen Bonus erhalten.

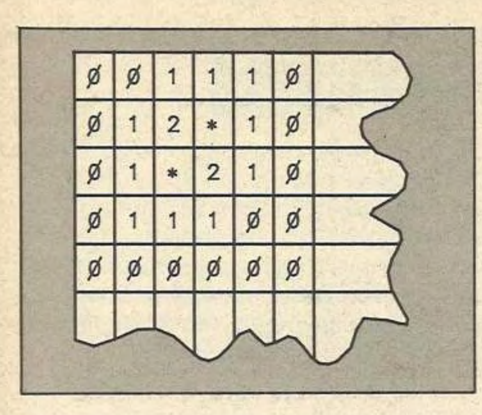


Bild 3. Für vollständig eingekreiste Minen gibt es bei der Punktezählung einen Bonus

Dann lädt Minefield die High-Score-Liste von Diskette (was ein paar Sekunden dauert) und sieht nach, ob Sie sich eintragen dürfen. Wenn ja, dann können Sie Ihren Namen (maximal acht Zeichen) und das Datum eintragen. Anschließend wird die neue High-Score-Liste auf Diskette verewigt.

Eingabehinweise

Das Programm arbeitet sehr häufig mit Steuerzeichen (siehe Listing 1). Bitte lesen Sie zuerst unsere Eingabehinweise auf Seite 159 und eventuell unseren Tippkurs auf Seite 157 durch, bevor Sie mit dem Abtippen beginnen.

Viel Spaß beim Spielen! (Gerti Noherr/tr)

```
1 REM MINEFIELD                                <160>
2 REM GESCHRIEBEN VON TR 2.10.86 FUER PLUS
  /4, C16 UND C116                             <149>
3 REM C64-VERSION TR 8.3.87                    <111>
5 DIM F%(31,16):POKE 53280,6:POKE 53281,1:
  POKE 650,128:POKE 788,52                     <066>
6 P(4)=.20:P(3)=.15:P(2)=.10:P(1)=.05          <085>
7 T$(1)="{RVSON,SPACE}TRAININGMODE{SPACE,R
  VOFF}"                                       <143>
8 T$(0)="{14SPACE}"                            <011>
10 PRINT CHR$(14)"{CLR,DOWN,RIGHT,GREEN}MI
   NEFIELD"                                    <114>
20 PRINT"{2RIGHT}ADAPTED FOR C64 BY TR"        <170>
30 PRINT"{2RIGHT}8.3.87"                       <120>
50 PRINT"{4DOWN,RIGHT,BLACK}SELECT LEVEL:"     <049>
51 PRINT"{RIGHT}*************"                 <234>
59 PRINT"{RIGHT}(4) SUICIDE                    <082>
60 PRINT"{RIGHT}(3) HARD                       <158>
61 PRINT"{RIGHT}(2) MEDIUM                     <090>
62 PRINT"{RIGHT}(1) EASY                       <201>
63 PRINT"{2DOWN,RIGHT}(T) TRAININGMODE"        <194>
64 PRINT"{RIGHT}(X) EXIT TO BASIC"             <020>
65 PRINT"{RIGHT}(C) CREATE NEW HIGHSCORE-F
   ILES                                        <062>
66 GOSUB 1100                                  <038>
70 GET K$:IF VAL(K$)>=1 AND VAL(K$)<5 THEN
    80                                         <193>
71 IF K$="X"THEN POKE 650,0:POKE 788,49:SY
   S 2048                                      <216>
72 IF K$="T"THEN TR=1-TR                       <037>
73 PRINT"{HOME,14RIGHT}"T$(TR)                 <226>
74 IF K$="C"THEN 3100                          <098>
75 GOTO 70                                     <045>
80 PRINT"{HOME,RIGHT,21DOWN}MOMPLS...":L=V
   AL(K$)                                      <189>
81 OPEN 1,8,15,"R:SCORES 1.MFD=SCORES 1.MF
   D":INPUT#1,A,B$,C,D:CLOSE 1                 <135>
82 IF A=63 THEN 85                             <188>
83 PRINT"{CLR,DOWN,RIGHT}THERE ARE NO HIGH
   SCORE-FILES ON THIS":PRINT"{RIGHT}DISK!
   !!":GOTO 1440                               <050>
85 AM=INT(P(L)*30*15):PRINT AM"{UP}"           <006>
90 FOR I=1 TO AM                               <017>
100 X=INT(RND(1)*30)+1                         <244>
110 Y=INT(RND(1)*15)+1                         <066>
115 IF F%(X,Y)=1 OR(X=2 AND Y=1)OR(X=30 AN
    D Y=14)OR(X=30 AND Y=15)OR(X=1 AND Y=1
    )THEN 100                                  <016>
120 F%(X,Y)=1:AM=AM-1:PRINT AM"{LEFT,SPACE
    ,UP}":NEXT                                 <201>
121 FOR X=0 TO 31:F%(X,0)=2:F%(X,16)=2:NEX
    T                                          <031>
122 FOR Y=1 TO 15:F%(0,Y)=2:F%(31,Y)=2:NEX
    T                                          <207>
130 PRINT"{CLR}"                               <118>
140 PX=1:PY=1:F%(30,15)=3:F%(1,1)=3            <209>
150 FOR Y=0 TO 14:POKE 1024+30+Y*40,93:NEX
    T                                          <166>
160 FOR X=0 TO 39:POKE 1024+X+15*40,64:NEX
    T:POKE 1024+30+15*40,113                   <014>
161 PRINT"{HOME,32RIGHT,DOWN}LEVEL";           <005>
162 PRINT L:POKE 1024+29+14*40,88:GOSUB 11
    00                                         <109>
163 PRINT"{HOME,27RIGHT,17DOWN,GREEN}MINEF
    IELD"                                      <064>
164 PRINT"{HOME,29RIGHT,18DOWN}BY TR{BLACK
    }"                                         <056>
165 IF TR=1 THEN GOSUB 1000                    <066>
170 GOSUB 2000:POKE 1024+(PX-1)+40*(PY-1),
    AZ+176                                     <001>
180 GOSUB 2200:IF F%(PX,PY)=1 THEN 1300        <174>
185 IF PX=30 AND PY=15 THEN 1500               <000>
186 IF F%(PX,PY)=0 THEN SC=SC+L:F%(PX,PY)=
    3                                          <039>
187 IF TR=0 THEN PRINT"{HOME,32RIGHT,3DOWN
    }"SC                                       <045>
190 GOSUB 2000:POKE 1024+(PX-1)+40*(PY-1),
    AZ+176                                     <021>
200 GOTO 180                                   <000>
1000 REM MINEN ANZEIGEN                        <207>
1005 FOR Y=1 TO 15:FOR X=1 TO 30               <137>
1010 IF F%(X,Y)=1 THEN POKE 1024+(X-1)+40*
     (Y-1),42                                  <080>
1020 NEXT X,Y                                  <187>
1030 IF TR=1 THEN PRINT"{HOME,8RIGHT,16DOW
     N}(TRAININGMODE)"                         <149>
1040 RETURN                                    <082>
1100 REM TASTENBELEGUNG                        <177>
1102 PRINT"{HOME,32RIGHT,7DOWN}KEYS:";         <168>
1104 PRINT"{DOWN,5LEFT}*****";                 <073>
1110 PRINT"{DOWN,5LEFT,RED}Q{2SPACE}W{2SPA
     CE}E";                                    <254>
1120 PRINT"{DOWN,7LEFT,3SPACE}-{3SPACE}";      <001>
1130 PRINT"{DOWN,7LEFT}A**+**D";               <133>
1140 PRINT"{DOWN,7LEFT,3SPACE}-{3SPACE}";      <021>
1150 PRINT"{DOWN,7LEFT}Z{2SPACE}X{2SPACE}C
     {BLACK}"                                  <082>
1200 RETURN                                    <242>
1300 REM AUF MINE GETRETEN                     <132>
1305 PRINT"{HOME,24DOWN}";                     <215>
1310 FOR I=1 TO 200                            <155>
1315 PRINT"{UP,10RIGHT}B O O M M M M !{SHI
     FT-SPACE}!{SHIFT-SPACE}!"                 <059>
1316 PRINT"{UP,RVSON,10RIGHT}B O O M M M M
      !{SHIFT-SPACE}!{SHIFT-SPACE}!"           <006>
1320 GET A$:IF A$=""THEN NEXT                  <185>
1334 POKE 1024+PX-1+40*(PY-1),170:POKE 198
     ,0                                        <201>
1335 IF TR=1 THEN 1360                         <155>
1340 PRINT"{HOME,RIGHT,23DOWN}HERE ARE THE
      REST OF THE MINES."                      <026>
1350 GOSUB 1000:POKE 1024+PX-1+40*(PY-1),1
     70                                        <015>
1360 PRINT"{HOME,RIGHT,23DOWN}PRESS <RETUR
     N>{17SPACE}"                              <242>
1370 GET K$:IF K$<>CHR$(13)THEN 1370           <124>
1420 GOSUB 2400:GOSUB 2500                     <024>
1440 PRINT"{2DOWN,RIGHT}PRESS <RETURN>"        <134>
1450 GET K$:IF K$<>CHR$(13)THEN 1450           <192>
1460 RUN                                       <232>
1500 REM GEWONNEN                              <167>
1510 PRINT"{HOME,RIGHT,18DOWN}YOU MADE IT!
     "                                         <232>
1552 IF TR=1 THEN GOSUB 2400:GOTO 1650         <108>
1553 GOSUB 2700:SC=SC+BS                       <133>
1554 PRINT"{HOME,33RIGHT,4DOWN,4SPACE}"        <224>
1555 PRINT"{HOME,32RIGHT,3DOWN}"SC             <122>
1561 GOSUB 2400                                <065>
1562 FOR I=1 TO 10:IF SC<HS(I)THEN NEXT:GO
     TO 1690                                   <197>
1563 FOR X=9 TO I STEP-1:HS(X+1)=HS(X):HN$
     (X+1)=HN$(X):HD$(X+1)=HD$(X):NEXT         <005>
1569 PRINT"{HOME,RIGHT,21DOWN}";               <123>
1570 INPUT"PLEASE ENTER YOUR NAME{4RIGHT}.
     .......{10LEFT}";HN$(I)                   <140>
1575 IF LEN(HN$(I))>8 THEN 1569                <126>
1576 IF LEN(HN$(I))<8 THEN HN$(I)=HN$(I)+"
     .":GOTO 1576                              <166>
```

```
1580 INPUT"{RIGHT}PLEASE ENTER DATE{5SPACE
     ,4RIGHT}ØØ.ØØ.ØØØØ{12LEFT}";HD$(I)      <Ø68>
1581 IF LEN(HN$(I))<8 THEN HN$(I)=HN$(I)+"
     .":GOTO 1581                            <Ø27>
1585 IF LEN(HD$(I))<>1Ø THEN PRINT"{2UP}":
     GOTO 158Ø                               <Ø76>
159Ø HS(I)=SC                                <Ø36>
1595 GOSUB 26ØØ                              <131>
16ØØ OPEN 1,8,15,"S:SCORES"+STR$(L)+".MFD"
     :CLOSE 1                                <Ø18>
161Ø OPEN 1,8,1,"SCORES"+STR$(L)+".MFD,P,W
     "                                       <238>
162Ø FOR I=1 TO 1Ø:GOSUB 18ØØ                <186>
163Ø PRINT#1,HX(I):PRINT#1,HY$(I):PRINT#1,
     HZ$(I)                                  <1Ø3>
164Ø NEXT:CLOSE 1                            <127>
165Ø GOSUB 25ØØ                              <17Ø>
166Ø PRINT"{2DOWN,RIGHT}PRESS <RETURN>"      <1ØØ>
167Ø GET K$:IF K$<>CHR$(13)THEN 167Ø         <182>
168Ø RUN                                     <198>
169Ø GOSUB 25ØØ                              <21Ø>
17ØØ PRINT"{DOWN}YOUR SCORE IS"SC            <Ø13>
171Ø GOTO 166Ø                               <158>
18ØØ HX(I)=64-HS(I):HY$(I)="":HZ$(I)=""      <Ø33>
181Ø FOR N=1 TO 8:HY$(I)=HY$(I)+CHR$(ASC(M
     ID$(HN$(I),N,1))+L+N):NEXT              <156>
182Ø FOR N=1 TO 1Ø:HZ$(I)=HZ$(I)+CHR$(ASC(
     MID$(HD$(I),N,1))+L+N):NEXT             <217>
183Ø RETURN                                  <11Ø>
19ØØ HS(I)=64-HX(I):HN$(I)="":HD$(I)=""      <Ø32>
191Ø FOR N=1 TO 8:HN$(I)=HN$(I)+CHR$(ASC(M
     ID$(HY$(I),N,1))-L-N):NEXT              <139>
192Ø FOR N=1 TO 1Ø:HD$(I)=HD$(I)+CHR$(ASC(
     MID$(HZ$(I),N,1))-L-N):NEXT             <Ø84>
193Ø RETURN                                  <21Ø>
2ØØØ REM BERECHNE ANZAHL MINEN               <254>
2Ø1Ø AZ=Ø                                    <24Ø>
2Ø2Ø IF F%(PX,PY-1)=1 THEN AZ=AZ+1           <187>
2Ø3Ø IF F%(PX+1,PY-1)=1 THEN AZ=AZ+1         <Ø91>
2Ø4Ø IF F%(PX+1,PY)=1 THEN AZ=AZ+1           <ØØ1>
2Ø5Ø IF F%(PX+1,PY+1)=1 THEN AZ=AZ+1         <Ø81>
2Ø6Ø IF F%(PX,PY+1)=1 THEN AZ=AZ+1           <221>
2Ø7Ø IF F%(PX-1,PY+1)=1 THEN AZ=AZ+1         <1Ø2>
2Ø8Ø IF F%(PX-1,PY)=1 THEN AZ=AZ+1           <13Ø>
2Ø9Ø IF F%(PX-1,PY-1)=1 THEN AZ=AZ+1         <154>
21ØØ RETURN                                  <126>
22ØØ REM STEUERUNG                           <211>
221Ø GET K$:IF K$=""THEN 221Ø                <216>
2215 POKE 1Ø24+(PX-1)+4Ø*(PY-1),AZ+48        <Ø22>
222Ø IF K$="W"AND PY>1 THEN PY=PY-1:RETURN   <1Ø7>
223Ø IF K$="E"AND PX<3Ø AND PY>1 THEN PX=P
     X+1:PY=PY-1:RETURN                      <152>
224Ø IF K$="D"AND PX<3Ø THEN PX=PX+1:RETUR
     N                                       <187>
225Ø IF K$="C"AND PX<3Ø AND PY<15 THEN PX=
     PX+1:PY=PY+1:RETURN                     <Ø54>
226Ø IF K$="X"AND PY<15 THEN PY=PY+1:RETUR
     N                                       <244>
227Ø IF K$="Z"AND PX>1 AND PY<15 THEN PX=P
     X-1:PY=PY+1:RETURN                      <247>
228Ø IF K$="A"AND PX>1 THEN PX=PX-1:RETURN   <139>
229Ø IF K$="Q"AND PX>1 AND PY>1 THEN PX=PX
     -1:PY=PY-1:RETURN                       <Ø64>
2291 IF K$="←"THEN GOSUB 29ØØ:GOTO 2296      <Ø53>
2292 POKE 5328Ø,Ø:FOR I=1 TO 1ØØ:NEXT:POKE
      5328Ø,6                                <Ø66>
2296 POKE 1Ø24+(PX-1)+4Ø*(PY-1),AZ+176       <163>
23ØØ GOTO 221Ø                               <Ø24>
24ØØ REM HIGHSCORES LADEN                    <142>
24Ø1 PRINT"{HOME,2ØDOWN,RIGHT}MOMPLS...{5S
     PACE}"                                  <236>
24Ø5 GOSUB 26ØØ                              <179>
241Ø OPEN 1,8,Ø,"SCORES"+STR$(L)+".MFD,P,R
     "                                       <Ø18>
242Ø FOR I=1 TO 1Ø:INPUT#1,HX(I):GOSUB 246
     Ø:GOSUB 19ØØ                            <Ø68>
245Ø NEXT:CLOSE 1:RETURN                     <132>
246Ø HY$(I)="":FOR N=1 TO 8:GET#1,A$:HY$(I
     )=HY$(I)+A$:NEXT:GET#1,A$               <ØØ2>
247Ø HZ$(I)="":FOR N=1 TO 1Ø:GET#1,A$:HZ$(
     I)=HZ$(I)+A$:NEXT:GET#1,A$:RETURN       <Ø19>
25ØØ REM HIGHSCORES ANZEIGEN                 <ØØØ>
25Ø5 PRINT"{CLR,DOWN,RIGHT}HIGHSCORES LEVE
     L"L                                     <123>
25Ø6 PRINT"{RIGHT}******************{DOWN}
     "                                       <Ø18>
251Ø FOR I=1 TO 1Ø:IF HS(I)=Ø THEN 253Ø      <124>
252Ø PRINT TAB(5-LEN(STR$(HS(I))))HS(I)" "
     HN$(I)"{2SPACE}"HD$(I)                  <184>
253Ø NEXT:RETURN                             <183>
26ØØ REM DISK-CHECK                          <184>
261Ø F$="SCORES"+STR$(L)+".MFD"              <123>
262Ø OPEN 1,8,15,"R:"+F$+"="+F$              <Ø34>
263Ø INPUT#1,A,B$,C,D:CLOSE 1                <Ø64>
264Ø IF A=63 THEN RETURN                     <186>
2641 IF A<>62 THEN PRINT"{CLR,DOWN,RIGHT}D
     ISK-ERROR:"A;B$;C;D:GOTO 2681           <1Ø2>
265Ø PRINT"{CLR,DOWN,RIGHT}INSERT DISK WIT
     H SCOREFILES!"                          <Ø68>
266Ø PRINT"{2DOWN,RIGHT}PRESS <RETURN>"      <Ø84>
267Ø GET K$:IF K$<>CHR$(13)THEN 267Ø         <168>
268Ø GOTO 262Ø                               <248>
2681 PRINT"{2DOWN,RIGHT}CHECK DRIVE AND PR
     ESS <RETURN>!":GOTO 267Ø                <131>
27ØØ REM BONUS FUER EINGEKREISTE MINEN       <Ø3Ø>
271Ø BS=Ø:FOR Y=1 TO 15:FOR X=1 TO 3Ø        <Ø89>
272Ø IF F%(X,Y)=1 THEN 274Ø                  <1Ø8>
273Ø NEXT X,Y:RETURN                         <Ø76>
274Ø IF F%(X-1,Y-1)=Ø THEN 273Ø              <Ø52>
275Ø IF F%(X,Y-1)=Ø THEN 273Ø                <ØØ2>
276Ø IF F%(X+1,Y-1)=Ø THEN 273Ø              <2ØØ>
277Ø IF F%(X+1,Y)=Ø THEN 273Ø                <2Ø7>
278Ø IF F%(X+1,Y+1)=Ø THEN 273Ø              <212>
279Ø IF F%(X,Y+1)=Ø THEN 273Ø                <Ø4Ø>
28ØØ IF F%(X-1,Y+1)=Ø THEN 273Ø              <1Ø4>
281Ø IF F%(X-1,Y)=Ø THEN 273Ø                <119>
283Ø POKE 1Ø24+(X-1)+4Ø*(Y-1),42             <Ø3Ø>
284Ø BS=BS+2*L                               <Ø41>
285Ø PRINT"{HOME,32RIGHT,4DOWN}"BS:GOTO 27
     3Ø                                      <Ø68>
29ØØ REM MARKIEREN                           <Ø86>
291Ø MX=PX:MY=PY                             <179>
292Ø P=1Ø24+(MX-1)+4Ø*(MY-1):I=PEEK(P):POK
     E P,171                                 <Ø11>
293Ø GET K$:IF K$<>""THEN 294Ø               <172>
2931 N=N+1:IF N=1Ø THEN POKE P,I             <Ø53>
2932 IF N=2Ø THEN POKE P,171:N=Ø             <Ø74>
2933 GOTO 293Ø                               <Ø71>
294Ø IF K$="W"AND MY>1 THEN MY=MY-1:GOTO 3
     Ø5Ø                                     <Ø12>
295Ø IF K$="E"AND MX<3Ø AND MY>1 THEN MX=M
     X+1:MY=MY-1:GOTO 3Ø5Ø                   <112>
296Ø IF K$="D"AND MX<3Ø THEN MX=MX+1:GOTO
     3Ø5Ø                                    <Ø55>
297Ø IF K$="C"AND MX<3Ø AND MY<15 THEN MX=
     MX+1:MY=MY+1:GOTO 3Ø5Ø                  <Ø33>
298Ø IF K$="X"AND MY<15 THEN MY=MY+1:GOTO
     3Ø5Ø                                    <112>
299Ø IF K$="Z"AND MX>1 AND MY<15 THEN MX=M
     X-1:MY=MY+1:GOTO 3Ø5Ø                   <Ø8Ø>
3ØØØ IF K$="A"AND MX>1 THEN MX=MX-1:GOTO 3
     Ø5Ø                                     <Ø44>
3Ø1Ø IF K$="Q"AND MX>1 AND MY>1 THEN MX=MX
     -1:MY=MY-1:GOTO 3Ø5Ø                    <Ø14>
3Ø2Ø IF K$="S"THEN 3Ø6Ø                      <Ø9Ø>
3Ø3Ø IF K$="←"THEN POKE P,I:RETURN           <Ø84>
3Ø35 POKE 5328Ø,Ø:FOR N=1 TO 1ØØ:NEXT:POKE
      5328Ø,6                                <127>
3Ø4Ø GOTO 293Ø                               <178>
3Ø5Ø POKE P,I:GOTO 292Ø                      <Ø1Ø>
3Ø6Ø IF I<>32 AND I<>35 THEN 3Ø35            <Ø57>
3Ø7Ø IF I=35 THEN POKE P,32:GOTO 292Ø        <255>
3Ø8Ø POKE P,35:GOTO 292Ø                     <247>
31ØØ REM NEUE HIGHSCORE-FILES ANLEGEN        <178>
31Ø5 PRINT"{CLR}MOMPLS..."                   <224>
311Ø OPEN 1,8,15,"R:SCORES 1.MFD=SCORES 1.
     MFD"                                    <148>
312Ø INPUT#1,A,B$,C,D:CLOSE 1:IF A=62 THEN
      32ØØ                                   <15Ø>
313Ø IF A<>63 THEN PRINT"{DOWN,RIGHT}DISK-
     ERROR:"A;B$;C;D:GOTO 144Ø               <11Ø>
314Ø PRINT"{DOWN,RIGHT}HIGHSCORE-FILES ALR
     EADY EXIST!"                            <166>
315Ø GOTO 144Ø                               <234>
32ØØ FOR L=1 TO 4                            <148>
321Ø OPEN 1,8,1,"SCORES"+STR$(L)+".MFD,P,W
     "                                       <Ø58>
322Ø FOR I=1 TO 1Ø:HS(I)=Ø:HN$(I)=".......
     .":HD$(I)="ØØ.ØØ.ØØØØ":GOSUB 18ØØ       <131>
323Ø PRINT#1,HX(I):PRINT#1,HY$(I):PRINT#1,
     HZ$(I)                                  <179>
324Ø NEXT:CLOSE 1:NEXT:RUN                   <233>
```

Listing 1. »Minefield«, bitte Eingabehinweise beachten

Fragen und Antworten

Der C64 von außen

Mich interessiert das Thema »Messen, Steuern, Regeln«. Welche Fähigkeiten in dieser Richtung besitzt der C64? (Dieter Maffei)

Für die Steuerung externer Geräte ist vor allem der User-Port geeignet. Der User-Port stellt acht »Ein-/Ausgabeleitungen« zur Verfügung. An jede einzelne der acht Leitungen kann eine Spannung von etwa fünf Volt angelegt und damit ein Elektrogerät (das allerdings nur einen sehr geringen Stromverbrauch besitzen darf!) ein- oder ausgeschaltet werden. Gesteuert wird der User-Port durch die CIA, einen Baustein des C64, der für die Ein-/Ausgabe zuständig ist. Durch das Setzen und Löschen einzelner Bits in verschiedenen CIA-Speicherzellen kann der Zustand der acht Leitungen gezielt beeinflußt werden. Umgekehrt kann ein Programm durch Auslesen dieser Speicherzellen abfragen, ob an einer der acht Leitungen eine Spannung anliegt oder nicht.

Der User-Port erlaubt nur die digitale Steuerung (Spannung angelegt/keine Spannung angelegt). Die Joystick-Ports enthalten unter anderem je zwei Analogeingänge. Mit diesen Analogeingängen können Änderungen (um genau zu sein: Widerstandsänderungen) in der »Umwelt« des C64 weitaus feiner abgestuft als mit dem User-Port erkannt werden. An diese Eingänge kann zum Beispiel ein lichtempfindlicher oder ein feuchtigkeitsempfindlicher Widerstand angeschlossen und der C64 als Licht- beziehungsweise Feuchtigkeitsmesser verwendet werden.

Druckerports

Was ist eigentlich der Unterschied zwischen dem Anschluß eines Druckers am seriellen Ausgang und dem Anschluß am User-Port? (Herbert Ziedler)

Zwischen User-Port und seriellem Ausgang gibt es beim C64 wesentliche Unterschiede. Ein Drucker, der mit dem seriellen Ausgang verbunden ist, kann ohne weiteres über die bekannten Basic-Befehle angesprochen werden (OPEN-Befehl, Geräteadresse 4). Die Daten werden dann seriell, also bitweise hintereinander, an den Drucker gesendet. Wird nun ein Drucker verwendet, der selbst eine Centronics-Schnittstelle besitzt, kann dieser nicht direkt seriell angeschlossen werden. Diese Drucker sind für parallelen Datenempfang eingerichtet (immer 8 Bit gleichzeitig). Um auch diese Drucker mit dem seriellen Ausgang des C64 zu verbinden, ist ein spezielles Gerät (Interface) notwendig, das die Daten von seriell nach parallel umwandelt. Es besteht aber auch die Möglichkeit, parallel arbeitende Drucker direkt an den C64 anzuschließen. Dazu wird der User-Port benötigt. Um den Drucker anzusprechen, werden dann ein spezielles Kabel und ein entsprechendes Programm benötigt, das die Daten auf den User-Port umlenkt. Viele kommerziell angebotenen Programme (Textverarbeitungen, Grafikprogramme) haben ein solches Steuerprogramm bereits eingebaut, so daß hier mit der Ansteuerung keine Probleme auftreten.

Datenfernübertragung

Ist es möglich, mit dem C64 Daten über das Telefonnetz zu übertragen? Kann man den C64 auf diese Weise mit anderen Computern verbinden? Teilen Sie mir bitte auch mit, welche Geräte dazu erforderlich sind und welche gesetzlichen Bestimmungen gelten. (Dirk Drechsler)

Ja, es ist ohne weiteres möglich, mit dem C64 Daten über das Telefonnetz zu übertragen (DFÜ = Datenfernübertragung). Sie benötigen dazu einen Akustikkoppler, der die Daten für das Telefonnetz aufbereitet und empfangene Daten wieder entschlüsselt. Zusätzlich benötigen Sie noch ein Terminalprogramm, das die Ansteuerung des Akustikkopplers übernimmt. Dann steht Ihnen die Welt der Mailboxen und öffentlichen Datenbanken offen. Sie brauchen nur noch die entsprechende Telefonnummer. Diese wird ganz normal gewählt. Sobald ein hoher Pfeifton ertönt, legen Sie den Hörer in die dafür vorgesehenen Muscheln des Kopplers, und schon kann es losgehen. Eine Alternative zu den Akustikkopplern sind die Modems. Bis auf die Postmodems darf allerdings keines dieser Geräte an das Telefonnetz angeschlossen werden. Zu beachten ist, daß nur öffentliche Datenbestände eingesehen werden dürfen. Der Versuch in eine nichtöffentliche Datenbank einzusteigen ist bereits strafbar.

Datenbanken selbstgemacht?

Welche Programmiersprache eignet sich am besten für die Programmierung einer leistungsfähigen Dateiverwaltung? Kann dazu auch Basic herangezogen werden? (Jörg Ullink)

Prinzipiell eignen sich verschiedene Programmiersprachen für Dateiverwaltungen. Da wäre natürlich als erstes das gute alte Basic. Allerdings stoßen Sie hier sehr schnell an die Grenzen des Interpreters im C64. Besser geeignet ist da schon das Basic des C128, welches über die nötigen Strukturanweisungen verfügt und leistungsfähigere Befehle zur Dateibehandlung bietet. Daneben gibt es natürlich noch so leistungsfähige Sprachen wie Pascal, mit denen sich Dateien beinahe optimal verwalten lassen, da alle nötigen Befehle im Sprachumfang enthalten sind. Mit allen bekannten Programmiersprachen entstehen jedoch meist nur sehr begrenzt anwendbare Dateiverwaltungssysteme. Viel flexibler zeigt sich hier eine programmierbare Datenbank wie etwa Superbase. Derartige Programme sind meist in Assembler geschrieben und speziell für ihr Anwendungsgebiet ausgelegt. Damit sind diese Programme meist schneller und auch komfortabler als eigene Entwicklungen.

Revolutionäre Sprache?

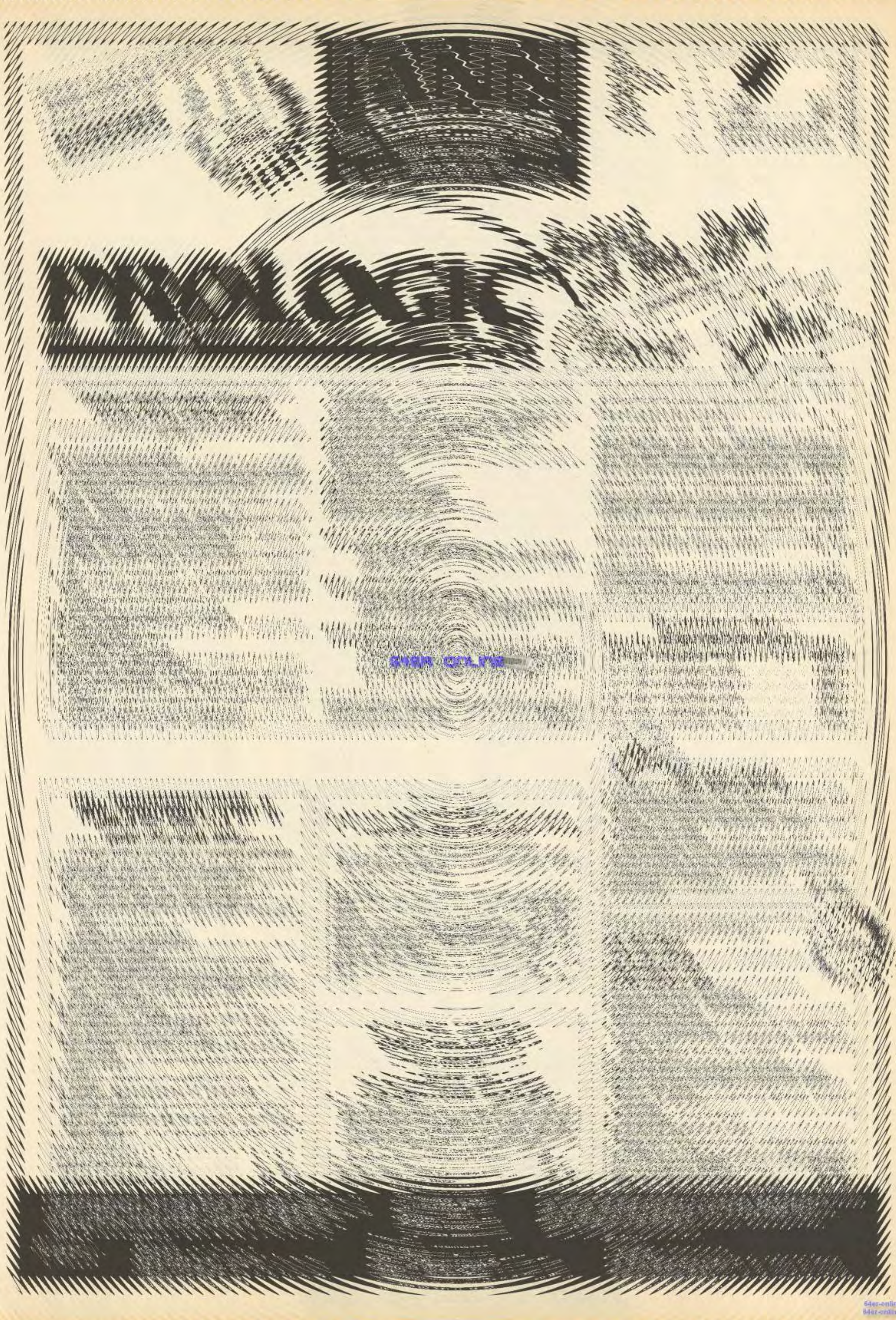
In letzter Zeit hört man immer wieder von der angeblich revolutionären Programmiersprache C. Wie unterscheidet sich C von herkömmlichen Sprachen? Ist C für den C64 erhältlich? (Hans Steppich)

Die Sprache C ist eine der jüngsten Sprachen auf dem Markt. Besondere Bedeutung erlangte die Sprache, als ein Betriebssystem unter Verwendung von C erstellt wurde (Unix). Vor allem wegen des modularen Aufbaus hebt sich C von anderen Sprachen ab. Die Sprache selbst besteht nur aus 28 Schlüsselwörtern. Diese reichen aus, um sich weitere Befehle selbst zu programmieren. Dadurch ist die Sprache sehr flexibel und kann auf nahezu jeden Anwendungsbereich zugeschnitten werden. Ähnlich wie in Pascal stehen auch die wichtigen Strukturanweisungen und Funktionen zur Verfügung. Ein weiterer Vorteil ist die für den Systemprogrammierer wichtige Betriebssystemnähe der Sprache. Mit einiger Übung kann C teilweise sogar den bisher oft unumgänglichen Assembler ersetzen. Daneben bietet natürlich auch C die Möglichkeit, Maschinenprogramme in das Programm einzubinden. Die für den C64 angebotenen C-Compiler halten sich zwar nicht an den von den Entwicklern vorgegebenen Standard, doch ist bei den meisten eine ausreichende Untermenge implementiert. Zum Teil unterstützen die Compiler für den C64 sogar die Musik- und Grafikprogrammierung, was ursprünglich nicht vorgesehen war. C ist heute bereits eine Sprache, die sich vor allem in Kreisen der Systemprogrammierer, aber langsam auch bei professionellen Anwendungen durchsetzt. Welche Compiler es für den C64 gibt, zeigt die Marktübersicht Programmiersprachen in Ausgabe 4/87 des 64'er-Magazins.

Kompatibilität

Was versteht man eigentlich unter dem Ausdruck »kompatibel«? Dieser Begriff wird zwar ständig gebraucht, aber nie erklärt. (Hans Duschke)

Kompatibilität ist eigentlich ein Begriff aus der Welt der Personal Computer. Seit jedoch der C128 auf dem Markt ist, findet sich dieser Ausdruck auch in der Welt der Heimcomputer. Ein Computer wird dann als kompatibel bezeichnet, wenn alle Programme eines anderen auch auf diesem Computer laufen. Dies ist beim C128 der Fall. Alle Programme des C64, bis auf die in allen Fällen auftretenden Ausnahmen, laufen



auch auf dem C128. Im Bereich der Software sind entsprechende Analogien zu finden. Erzeugt beispielsweise eine Textverarbeitung Dateien, die auch von einer anderen verwendet werden können, kann man ohne weiteres behaupten, die beiden Produkte sind zueinander kompatibel. Genau ins Deutsche übersetzt bedeutet kompatibel soviel wie »zusammenpassend«.

Spezielle Roboter

Ich habe schon viel von »Teach-in-Robots« gehört. Was versteht man darunter, und wie steuert man einen solchen Roboter mit einem Heimcomputer?

(K.-H. Wender)

Ein »Teach-in-Robot« ist ein Roboter, der in Grenzen lernfähig ist. Er merkt sich Bewegungen, die ihm vorgeführt werden und führt später die gleichen Bewegungen »selbständig« aus. Ein Beispiel: Sie bauen ein Modell, das über einen in allen Richtungen beweglichen Greifarm verfügt. Am Ende des Arms befindet sich ein Elektromagnet. Das Modell (der »Roboter«) soll von einem Münzenstapel Münze für Münze abheben und auf ein »Fließband« legen.

Um dem Roboter seine Aufgabe beizubringen, verwenden Sie den »Lernmodus«: Sie steuern den Roboterarm »per Hand« (zum Beispiel mit den Cursor-Tasten) und positionieren ihn auf dem Münzenstapel. Jede Bewegung des Arms dreht die Achsen von Potentiometern, die an den Heimcomputer angeschlossen sind (an den Joystick-Ports des C64). Ihr Programm merkt sich nun die aktuellen Potentiometerwerte, die die Armposition definieren – Sie haben dem Roboter beigebracht, wo sich der Münzenstapel befindet.

Auf die gleiche Weise steuern Sie den Arm zum Fließband. Das Programm merkt sich erneut die aktuellen Potentiometerwerte und kennt nun die Position des Fließbands. Im »Ausführungsmodus« steuert das Programm den Arm zur ersten gespeicherten Position (dem Münzenstapel), schaltet den Elektromagneten ein, und eine Münze wird angezogen. Anschließend wird der Arm zur zweiten Position bewegt und der Elektromagnet ausgeschaltet – die Münze fällt auf das Fließband. Der Roboter führt die Bewegungen selbständig aus.

Druckerschalter

In Ihren Druckertests lese ich ständig etwas über DIP-Schalter, kann mir aber nicht vorstellen, was diese Schalter genau für eine Bedeutung haben.

(Michael Zehetner)

Jeder Anwender kennt eigentlich die Probleme, die mit Druckern in Verbindung mit verschiedenen kommerziellen Programmen auftreten. Der Drucker vollführt hier mal einen unbeabsichtigten Seitenvorschub, druckt keine Umlaute oder bringt den Ausdruck mit doppeltem Zeilenabstand zu Papier. Um nun diese Unannehmlichkeiten zu beseitigen, braucht der Anwender eine Möglichkeit, den Drucker entsprechend anzupassen. Diese Anpassung läßt sich teilweise durch die DIP-Schalter erreichen. Dabei handelt es sich um kleine Schalterchen, deren Stellung die Arbeit des Druckers direkt beeinflussen. Bei modernen Druckern kann hier vom Zeichensatz bis hin zur verwendeten Papierart (endlos oder Einzelblatt) jede Menge eingestellt werden. Sendet beispielsweise ein Textverarbeitungsprogramm nach jeder Zeile einen Zeilenvorschub, wird dieser per DIP-Schalter am Drucker abgeschaltet, da dies ja bereits vom Programm aus erfolgt. Auf diese Weise läßt sich ein Drucker an die verschiedensten Programme durch bestimmte Schalterstellungen ohne weiteres anpassen und wird somit zu einem flexiblen Ausgabegerät.

Welcher Monitor?

Wo liegt der Unterschied zwischen einem Maschinensprache- und einem Diskettenmonitor? Ich schaffe es einfach nicht, mit einem Diskettenmonitor zu programmieren, wie es mit einem entsprechenden Maschinensprachemonitor ohne weiteres möglich ist.

(Herbert Lünger)

Bei Maschinensprache- und Diskettenmonitoren handelt es sich um zwei prinzipiell verschiedene Programme. Wohlgemerkt, Monitore in diesem Sinne sind nichts anderes als Maschinensprache-Programme, die den Anwender befähigen, einen Einblick in die Speicherlandschaft des Computers oder der Diskette zu nehmen. Damit ist auch der große Unterschied zwischen den beiden Monitoren bereits angesprochen. Mit einem Maschinensprache-Monitor können Sie sich den Speicher des C64 an beliebigen Stellen betrachten und auch verändern. Ein derartiges Programm erlaubt auch die direkte Maschinensprache-Programmierung. Allerdings verliert man hierbei schnell die Übersicht, für kurze Routinen ist der Monitor aber ohne weiteres geeignet. Beim Diskettenmonitor erfolgt der Zugriff nicht auf den Speicher, sondern direkt auf die momentan eingelegte Diskette. Damit lassen sich, wie auch beim Maschinensprache-Monitor, die Inhalte des Datenträgers in Form von Hexadezimal-Zahlen auf dem Bildschirm darstellen. Gute Diskettenmonitore sind sogar in der Lage, zusätzlich zur hexadezimalen Darstellung die entsprechenden Zeichen auf den Bildschirm zu bringen. Sogar die Befehle von Basic- oder Maschinensprache-Programmen werden mit angezeigt. Eine direkte Programmierung ist damit nicht unbedingt möglich. Natürlich können Sie die Werte, die auf Diskette gespeichert sind, verändern. Hier ist jedoch äußerste Vorsicht angebracht, da sehr schnell für die Diskettenorganisation wichtige Daten zerstört werden, so daß die Diskette ohne genaue Kenntnisse über deren Aufbau nicht mehr verwendet werden kann.

CP/M-Modul

Ich habe für den C64 ein CP/M-Modul erworben. Nun würde mich interessieren, ob es damit möglich ist, dBase II, wie es für den C128 angeboten wird, damit ablaufen zu lassen. Laufen auch die anderen angebotenen CP/M-Programme mit diesem Modul? (Dieter Hülsner)

Informieren Sie sich als erstes über die vorliegende Versionsnummer des Betriebssystems. Davon hängt die Einsetzbarkeit von CP/M-Programmen zu erheblichen Teilen ab. Die bekannten Module für den C64 arbeiten mit der CP/M-Version 2.2. Damit scheidet die dBase-II-Version, die für den C128 angeboten wird, aus, da diese für CP/M 3.0 gedacht ist. Für CP/M 2.2 gibt es auch leistungsfähige Produkte. Allerdings treten Probleme mit den verwendeten Diskettenformaten auf. Während das Floppylaufwerk 1571 beim C128 ohne weiteres in der Lage ist, verschiedene Formate zu verarbeiten, sind Sie beim C64 auf das Format der 1541 angewiesen. Das wiederum bedeutet, daß die gewünschten Programme auf einer Diskette mit genau diesem Format vorliegen müssen. Zur Zeit ist kein Vertreiber bekannt, der CP/M-2.2-Programme auf diese Weise anbietet.

Computer oder Schreibmaschine?

Welche Vorteile hat eine Textverarbeitung gegenüber der herkömmlichen Schreibmaschine, außer der ständigen Verfügbarkeit des Textes? Welche zusätzlichen Einrichtungen werden benötigt, um ein solches Programm effektiv nutzen zu können?

(Gerd Dörfner)

Eine Textverarbeitung besitzt gegenüber einer herkömmlichen Schreibmaschine sicherlich mehr Vorteile als die Wiederverwendbarkeit eines Textes. So kennen Sie sicherlich das Problem mit den leidigen Tippfehlern. Entweder man korrigiert diese mit Tipp-Ex, was nicht zum Vorteil des Dokuments ausfällt, oder Sie schreiben das Ganze noch einmal von vorne. Eine Textverarbeitung enthebt Sie dieser Last, da es ein derartiges Programm erlaubt, Korrekturen von Tipp- oder Formulierungsfehlern direkt am Bildschirm vorzunehmen. Auch das Löschen oder Einfügen beliebiger Texte ist problemlos möglich. Floskeln, wie etwa Anreden oder Grußformeln, ja sogar ganze Textbausteine werden nur einmal geschrieben und dann über einfache Befehle oder Tastenkombinationen immer wieder von neuem von Diskette in den bestehenden Text eingefügt. Richtig ausnützen kann man die leistungsfähigen Programme natürlich erst, wenn der richtige Drucker zur Verfügung steht. Je nach Ausstattung kann der geschriebene Text auch optisch verfeinert werden, etwa durch Fettschrift, Kursivdruck, Unterstreichungen, Schmalschrift etc. Auch sind Sie in der Lage Serienbriefe zu drucken, wobei Sie festlegen können, wie oft der Brief gedruckt oder ob Textbausteine während des Druckens eingefügt werden sollen. Sie sehen also, daß eine Textverarbeitung im Zusammenspiel mit den entsprechenden Zusatzgeräten die Schreibmaschine nicht nur ersetzen, sondern auch bei weitem übertreffen kann, was die Leistung angeht.

Computer-Lexikon

Oft gesucht und selten gefunden – eine ausführliche Übersicht über die wichtigsten und gebräuchlichsten Begriffe rund um den Computer.

Akkumulator (engl. Accumulator)
Ein spezielles Register eines Prozessors, in dem die meisten logischen Rechenoperationen durchgeführt werden.

Akustikkoppler (engl. Acoustic coupler)
Ein Akustikkoppler (Bild 1) ist eine spezielle Variante eines Modems. Wie auch das Modem wandelt der Akustikkoppler binäre (Computer-)Signale und Informationen in elektroakustische Signale um und umgekehrt. Der Akustikkoppler ist dabei zwischen Computer und Telefonnetz geschaltet, wobei die Verbindung zur Telefonleitung über den Telefonhörer hergestellt wird. Hör- und Sprechmuschel des Telefonhörers werden schallgedämpft in die Gummimanschetten des Kopplers gelegt. In den Manschetten befinden sich ein Mikrofon und ein Lautsprecher, über die akustische Signale empfangen und gesendet werden. Die Datenübertragung mittels Akustikkoppler ist wegen der zusätzlichen elektrisch-akustisch-elektrischen Umwandlung wesentlich störanfälliger als bei einem Modem.

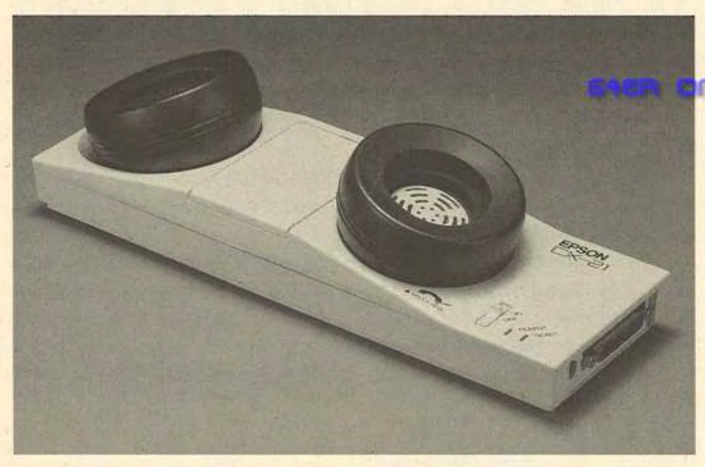


Bild 1. Ein handelsüblicher Akustikkoppler

Algorithmus (engl. algorithm)
Es gibt verschiedene Methoden, eine Aufgabe zu lösen. Ein Algorithmus ist ein Verfahren zur Lösung eines Problems, das nach endlich vielen Schritten abbricht und dabei entweder eine Lösung des Problems produziert hat oder das Problem als unlösbar zurückweist.

Auch wenn Algorithmen als »effektive Verfahren« bezeichnet werden, müssen sie keineswegs immer effizient sein. Ein Beispiel-Algorithmus zum Gedichteschreiben etwa könnte alle erdenklichen Kombinationen aus Buchstaben und Satzzeichen bis zu einer vorgegebenen Länge von 500 Zeichen bilden. Die Anzahl der Möglichkeiten ist zwar ungeheuer groß, aber endlich. Alle Möglichkeiten könnten unter erheblichem Aufwand mit einem solchen Algorithmus erzeugt werden. Man erhält dadurch zwar jede Menge Unsinn, aber garantiert auch alle möglichen Gedichte mit 500 Zeichen Länge.

In der EDV charakterisiert man diese Methode als »brute-force« (etwa: mit roher Kraft).

Adresse (engl. address)
Die Adresse gibt an, an welcher Stelle sich bestimmte Daten in einem Computerspeicher befinden (vergleichbar etwa mit Hausnummern in einer langen Straße). Die Speicherplätze sind durchnumeriert, wodurch ihnen Adressen zugeteilt werden, über die auf den Inhalt zugegriffen werden kann.

Alphanumerische Zeichen (engl. alphanumeric)
Darunter versteht man alle Zeichen, die ein Computer auf einer Ausgabeeinheit (zum Beispiel Bildschirm oder Drucker) darstellen kann; zumindest aber das Alphabet und die Dezimalzahlen.

ALU (engl. arithmetic logic unit)
Die »arithmetisch-logische Einheit« ist der für die Durchführung sämtlicher Rechenoperationen verantwortliche Teil eines Prozessors.

Anweisung (engl. statement)
Ein einzelner oder zusammengesetzter Ausdruck innerhalb eines Programms, der dem Computer mitteilt, was er ausführen soll.

Anwendungsprogramm (engl. applications program)
Ein »Dienstleistungsprogramm« mit meist speziellem Anwendungsbereich (Textverarbeitung, Datenverwaltung), im Gegensatz zu einem Systemprogramm, mit dessen Hilfe unter anderem auch Anwendungsprogramme erstellt werden können.

ASCII (engl. american standard code for information interchange)
Ein Standardcode, der 128 Zeichen (Zahlen, Buchstaben, Symbole, Sonderzeichen) ein aus sieben Bit bestehendes Muster zuweist. Dieser Standard erleichtert die Kommunikation verschiedener Programme, Computer und Peripheriegeräte (Drucker).

Assembler
1. Eine maschinenorientierte Programmiersprache, die von Prozessor zu Prozessor unterschiedlich ist. Assemblerprogramme können nur zusammen mit »ihrem« Prozessortyp laufen, im Gegensatz zu den höheren Programmiersprachen, die unabhängig von dem eingesetzten Mikroprozessor sind. Assemblerprogramme nutzen die Fähigkeit des entsprechenden Prozessors optimal aus, auch wird – im Vergleich etwa zu Basic – nur sehr wenig Speicherplatz belegt und die Programme sind zudem sehr schnell. Aus diesem Grund werden Assemblerprogramme bei zeitkritischen Prozessen auch in Programme, welche in einer höheren Programmiersprache geschrieben sind, eingebunden.

2. Übersetzungsprogramm, welches ein in Assemblercode vorliegendes Programm (Quellcode) in den binären, also direkt ablauffähigen Maschinencode umwandelt. Dabei werden die Assemblerbefehle (auch »Mnemonics« oder »Opcodes« genannt) auf syntaktische Fehler überprüft.

Auflösung (engl. resolution)
Mit »Auflösung« oder »Bildauflösung« bezeichnet man die maximale Anzahl optisch voneinander unterscheidbarer Punkte, die auf einer Fläche dargestellt werden können.

Angegeben wird die Auflösung in Pixels. Pixel ist ein Kunstwort aus dem englischen »picture element«, zu deutsch »Bildpunkt«.

Grafikcomputer höchster Auflösung können 4096 x 4096 Pixels darstellen, im Bereich der Heimcomputer sind etwa 320 x 160 Pixels der Standard.

BAM (engl. block availability map)

Die BAM ist ein Verzeichnis der freien und belegten Sektoren (Blöcke) auf einer Diskette. Für jeden Block ist ein Bit reserviert. Steht dieses Bit auf logisch 1, so ist der entsprechende Block frei. Wird nun eine Datei auf die Diskette geschrieben, so belegt diese natürlich einen oder mehrere Blöcke. Damit das DOS weiß, welche Blöcke auf einer Diskette belegt sind, werden diese in der BAM als belegt gekennzeichnet, indem das entsprechende Bit auf 0 gesetzt wird.

Die BAM steht bei den Commodore-Laufwerken 1541 und 1570 auf Spur 18, Sektor 0; bei der 1571 auf den Spuren 18 und 53, Sektor 0. Es wird nur etwa die Hälfte des Sektors von der BAM belegt, der andere Platz wird unter anderem für Diskettenname, Formatkennung etc. benutzt.

BDOS, BIOS, CCP

Aus diesen drei Grundelementen setzt sich CP/M zusammen. Das BDOS (Basic Disk Operating System) besteht aus einer Vielzahl einzelner Funktionen, welche den Datenfluß zwischen dem eigentlichen Anwenderprogramm und den verschiedenen Ein-/Ausgabegeräten steuern. Das BIOS (Basic Input/Output System) übernimmt die Daten vom BDOS und leitet den Datentransfer in die Wege. Der CCP (Control Command Processor) ist für die Eingaben des Anwenders zuständig. Er stellt unter anderem fest, ob ein residenter oder transienter Befehl aufgerufen wurde (Zur Ausführung eines transienten Befehls wird vor der Ausführung die entsprechende .COM-Datei von Diskette geladen).

Das Wort »Basic« im Zusammenhang mit BDOS und BIOS hat nichts mit der gleichnamigen Programmiersprache zu tun, sondern steht für »grundlegend«.

Befehl (engl. instruction)

Kommandoanweisung, die einen Computer bestimmte Operationen ausführen läßt. Auch Zuweisungen sind Befehle. So ist bei der Basic-Zuweisung »A=7x5« die rechte Seite »7x5« ein Ausdruck, dessen Wert (35) der Variablen »A« zugewiesen wird.

Es gibt auch Programmiersprachen, die gänzlich ohne Befehle auskommen, sogenannte »funktionale Sprachen«, welche nur Ausdrücke kennen, wie zum Beispiel Lisp.

Betriebssystem (engl. operating system)

Ein Programm, das die Fähigkeiten eines Computers so organisiert, daß Anwendungsprogramme lauffähig und damit benutzbar werden. Das Betriebssystem zählt zur Systemsoftware. Es befindet sich entweder bereits fest eingebaut im Computer-ROM oder wird bei jedem Einschalten des Systems von Diskette geladen.

Binär (engl. binary)

Ein Zahlensystem mit der Basis zwei, bei dem die Zahlen nur durch Kombinationen der Ziffern 0 und 1 dargestellt werden. Da sich diese Zahlen leicht in einem Entweder-Oder-Schema darstellen lassen (Strom fließt, Strom fließt nicht), wurde das binäre Zahlensystem zum Charakteristikum digitaler Computersysteme.

Bit (engl. binary digit)

Kleinste informationstragende Darstellungseinheit im binären Zahlensystem. Das Binärzeichen kann nur zwei Zustände annehmen, die meist mit 0 (für »falsch«, »nein«, »kein Strom«) und 1 (»wahr«, »ja«, »Strom«) bezeichnet werden.

Bit/s

Bit pro Sekunde ist eine Maßeinheit für die Geschwindigkeit, mit der Daten übertragen werden. Oft spricht man in diesem Zusammenhang auch von »Baud« beziehungsweise der »Baud-Rate«. Baud ist aber eine veraltete Maßeinheit.

Die Ausgabe in bit/s gibt nur Auskunft über die Geschwindigkeit des Informationsflusses und darf nicht mit der Anzahl der übertragenen Zeichen gleichgesetzt werden, da sich ein Zeichen aus mehreren Bits zusammensetzt. Außer den 7 oder 8 Datenbit, die benötigt werden, um ein Zeichen darzustellen, müssen noch ein Startbit und ein oder zwei Stoppbits sowie unter Umständen ein Paritätsbit übertragen werden. Als Faustregel gilt: Tatsächliche Übertragungsrate = (bit/s) / 10. Kürzel geben Auskunft über die zur Übertragung eines Zeichens notwendigen Bits; so bedeutet »8n1« zum Beispiel 8 Datenbits, kein Paritätsbit und 1 Stoppbit.

Bus

Datenübertragender Kommunikationspfad zwischen den verschiedenen Bestandteilen eines Computersystems. Ein Bus kann verborgen sein (beispielsweise als Bestandteil einer integrierten Schaltung) oder frei zugänglich, um Peripheriegeräte oder Erweiterungsplatinen aufzunehmen. Besonders für diesen Zweck ist ein Bus in Mikrocomputersystemen von besonderer Bedeutung, da durch ihn eine einheitliche Schnittstelle für die Systemerweiterungen definiert wird.

Byte, KByte

Ein Byte ist eine aus 8 Bit zusammengesetzte Informationseinheit, mit der sich jeder Wert zwischen 0 und 255 darstellen läßt.

Ein »Wort« ist aus zwei, ein Langwort (»long word«) aus vier Byte zusammengesetzt; ein Nibble umfaßt ein halbes Byte (4 Bit).

Ein KByte besteht aus 1024 Byte.

Der C64 besitzt 64 KByte RAM (das sind 65536 = 64 x 1024 Byte).

Centronics-Schnittstelle

Parallele Schnittstelle (Bild 2) speziell zum Anschluß von Druckern. Diese von der Firma Centronics entwickelte Schnittstelle wurde von vielen Herstellern übernommen und kann somit als inoffizielle Norm bezeichnet werden. Die Centronics-Schnittstelle zeichnet sich vor allem durch relativ geringen Hardwareaufwand aus, da zur Verbindung von zwei Geräten, an denen Centronics-Anschlüsse beste-

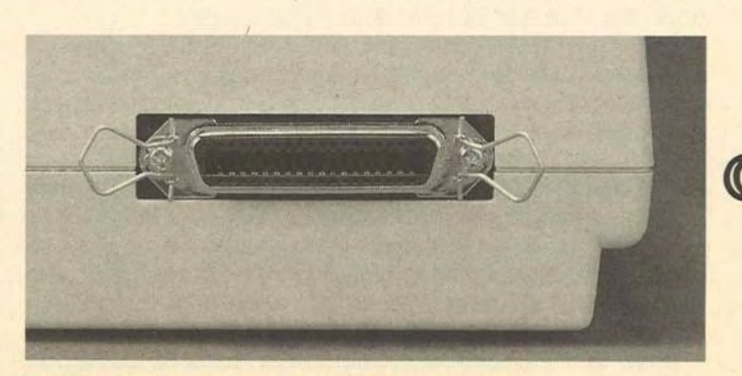


Bild 2. Die Centronics-Parallel-Schnittstelle

hen, nur ein 36poliges Kabel mit entsprechenden Steckern benötigt wird.

Chip
Ein Chip ist ein rechteckiges Siliziumplättchen, welches nach diversen Herstellungsprozessen bestimmte elektrische Funktionen erfüllt. In Computern findet man diese Chips in den integrierten Schaltkreisen (ICs), wo sie die verschiedensten Funktionen erfüllen können (Mikroprozessoren, Speicher- und Logikbausteine, Bild 3).

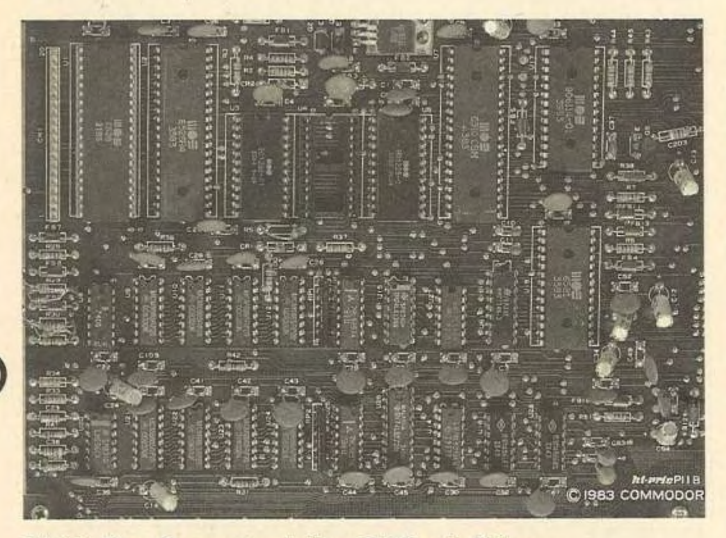


Bild 3. Eine Computerplatine (C64) mit Chips

Compiler (Übersetzer)
Ein Computerprogramm, das ein in einer Hochsprache (wie Basic oder Pascal) geschriebenes Programm in Maschinensprache übersetzt, das heißt, in das Befehlsformat des Mikroprozessors umwandelt.
Solche »compilierten Programme« sind in der Ausführung wesentlich schneller als »interpretierte«.

CP/M (engl. Control Program for Microcomputers)
CP/M ist ein diskettenorientiertes Betriebssystem und wurde 1975 von Gary Kildall/Digital Research für Computer mit Intel 8080-Prozessoren entwickelt. Heutzutage wird in CP/M-Computern fast ausschließlich der Zilog-Z-80-Prozessor verwendet, wie auch im Commodore C128. CP/M ist leicht auf verschiedenen Geräten zu installieren, was durch Trennung aller computerspezifischen Routinen von der eigentlichen Verwaltungslogik erreicht wird. Hardwareabhängige Teile sind im BIOS untergebracht, der logische Teil des Systems befindet sich im BDOS. Innerhalb einer CP/M-Version (zum Beispiel CP/M 3.0) ist das BDOS auf allen Computern identisch, das BIOS jedoch hardwareabhängig. Soll CP/M auf einem anderen Computer mit Z-80-Prozessor installiert werden, so ist lediglich das BIOS entsprechend abzuändern.

Datei (engl. file)
Eine Datei ist eine Sammlung von zusammengehörenden Informationen, die gemeinsam gespeichert sind. Für den Computer stellt eine Datei nichts anderes als eine Verwaltungseinheit dar. Der Computer legt die Datei in einem bestimmten Format auf Diskette beziehungsweise Kassette an. CP/M-Dateien auf einer Diskette bestehen beispielsweise aus Datensätzen (»Records«) zu je 128 Bytes, Dateien auf dem 1541-Laufwerk von Commodore stehen in 256-Byte-Blöcken (»Blocks«) auf der Diskette.
Eine für den Benutzer wichtige Unterscheidung der Dateien ergibt sich aus der Art des Zugriffs, der auf die gespeicherten Informationen möglich ist. Man unterscheidet dabei hauptsächlich zwischen sequentiellem und direktem Zugriff beziehungsweise sequentiellen und relativen Dateien.

Datei, relative
Bei der relativen Datenspeicherung wird davon ausgegangen, daß jede Datei aus vielen Einträgen besteht, die alle eine gewisse Maximallänge nicht überschreiten. Für jeden Eintrag, dessen Größe der Benutzer vorher definiert, wird ein eigener Datensatz (»Record«) angelegt, auf den bei späterer Bearbeitung direkt zugegriffen werden kann. Oft wird zusätzlich in einer sequentiellen Datei, der sogenannten »Indexdatei«, festgehalten, wo (physikalisch) ein bestimmter Datensatz auf der Diskette zu finden ist. Auf diese Weise kann man sehr schnell auf jeden Datensatz zugreifen, denn die Indexdatei kann ständig im Computer-RAM verbleiben. Außerdem kann so recht schnell sortiert werden, denn man braucht ja nicht die eigentlichen Daten, sondern nur die Indexdatei nach den gewünschten Kriterien zu sortieren.

Datei, sequentielle
Organisationsform für Dateien, bei der die einzelnen Datensätze auf dem Speichermedium hintereinander abgelegt sind. Diese Methode ist immer dann angezeigt, wenn die Datei Informationen enthält, die stets in der gleichen Reihenfolge (von »vorne« nach »hinten«) gelesen werden müssen (zum Beispiel Programme). Der Nachteil der sequentiellen Datei besteht darin, daß man nicht auf einen bestimmten Eintrag willkürlich zugreifen kann, sondern solange »der Reihe nach« suchen muß, bis man den Eintrag gefunden hat. Das Auffinden weiter »hinten« liegender Daten dauert natürlich länger als die weiter »vorne« liegender. Außerdem ist bei diesem Dateityp ein Schreiben neuer Informationen nur hinter dem Ende der zuletzt gespeicherten Daten möglich. Sollen die Daten auf Band (Datasette) geschrieben werden, ist nur die sequentielle Organisationsform möglich.

Datex-P
Datex-P ist ein Datenübertragungsnetzwerk, welches 1980 von der Deutschen Bundespost eingerichtet wurde. Bedingt durch die spezielle Art der Datenübermittlung (Paketvermittlung, daher auch das »P« hinter Datex) ist die Übertragung hier günstiger als über das Telefonnetz.
Datex-P nimmt die notwendigen Anpassungen der Computer untereinander selbst vor, dies betrifft besonders die Übertragungsgeschwindigkeit.
Für Datex-P benötigt man neben der »üblichen« DFÜ-Ausrüstung (Computer, Terminalprogramm, Akustikkoppler, Telefon) noch eine Benutzerkennung für Datex-P (»NUI«, Network User Identification) und die Nummer des gewünschten Computers im Datex-P-Netz (»NUA«, Network User Adress). Die NUI kann bei der Post beantragt werden und kostet 15 Mark im Monat Grundgebühr zuzüglich Kosten der Verbindungen.

DFÜ
Unter DFÜ (Abkürzung für Datenfernübertragung) versteht man die Übermittlung von Daten über weite Entfernungen. Dies kann über das Telefonnetz geschehen, aber auch drahtlos (Beispiel: RTTY, Funkfernschreiben). Über das Telefonnetz stehen gleich eine ganze Reihe von Möglichkeiten zur Verfügung: Datex und Bildschirmtext, zwei Dienste der Deutschen Bundespost, und natürlich jede Menge elektronischer Briefkästen, sogenannte Mailboxen, welche meist privat, teilweise aber auch kommerziell betrieben werden. Die Hobby-DFÜ mit Heimcomputern hat in letzter Zeit viele Freunde gewonnen.

Um DFÜ zu betreiben, muß der Anwender seinen Computer mit dem Übertragungsmedium, also meist mit dem Telefonnetz, verbinden. Hierzu gibt es spezielle Geräte, sogenannte »Modems« und »Akustikkoppler« sowie spezielle Steuersoftware (Terminalprogramme).

Dialog
Mit Computern kann man auf zwei verschiedene Arten kommunizieren: im Stapel-Verfahren und im Dialogverfahren.

Beim Stapelverfahren gibt der Anwender die notwendigen Daten hintereinander ein und erhält nach einiger Zeit das Ergebnis. Dieses Verfahren bezeichnet man auch als Batch-Betrieb.

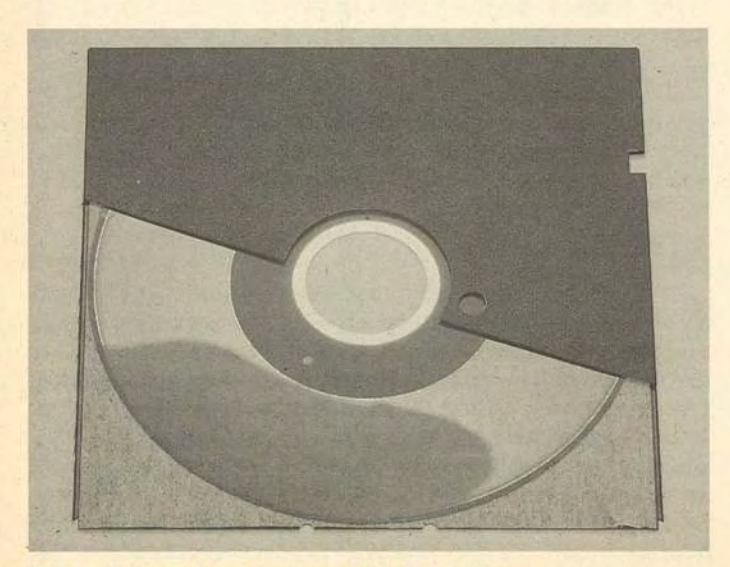
Beim Dialog-Verfahren reagiert das System sofort auf die einzelne Dateneingabe, das heißt, es findet ein ständiger Wechsel zwischen Eingabe (durch den Anwender) und Ausführung (durch den Computer) statt. Der Dialog wird dabei durch das Programm gesteuert.

Directory (Disketteninhaltsverzeichnis)
Einer der vielen Vorteile der Diskette gegenüber der Kassette besteht darin, daß sie ein Inhaltsverzeichnis enthält.

Dieses Verzeichnis ermöglicht es, festzustellen, wie viele und welche Dateien auf der Diskette enthalten sind, wieviel Platz sie benötigen und ob noch Platz für weitere Dateien ist.

Direktzugriff
Der Zugriff (engl. direct access) ist die Art, in der einzelne Speicherzellen eines Datenträgers »aufgesucht« werden. Bei Commodore-Diskettenlaufwerken versteht man unter Direktzugriff die Möglichkeit, auf einen bestimmten Block der Diskette direkt zugreifen zu können, ohne die Datei, in der der Block steht, vorher laden zu müssen. Das eingebaute DOS der Commodore-Laufwerke stellt eine ganze Reihe von Befehlen zur Verfügung, mit denen ein direkter Zugriff verhältnismäßig einfach zu realisieren ist. Beim Programmieren muß man jedoch sehr sorgfältig vorgehen, da auf DOS-Ebene die Befehle keiner Überprüfung unterliegen: Gibt man etwa die Anweisung, auf Spur 90 etwas zu lesen oder zu schreiben, so versucht das DOS auch, den Schreib-/Lesekopf des Diskettenlaufwerks dort zu positionieren, obwohl gar keine Spur 90 existiert.

Diskette
auch »Floppy-Disk« genannt. Floppy heißt im Englischen soviel wie »schlaff« oder »schlotterig«. Eine Diskette (Bild 4) ist ein sehr preisgünstiges Speichermedium, das aus einer flexiblen (daher: »floppy«) Kunststoffscheibe mit magnetisierbarer Oberfläche besteht. Eine steife Hülle aus Plastik gibt der Diskette Stabilität und schützt ihre Oberfläche. Die Hülle läßt lediglich die für die Funktion notwendigen Bereiche (Schreib-/Leseöffnung, Indexloch) frei. Die im Heimcomputerbereich üblichen Formate sind 5¼ und 3½ Zoll. Bei Disketten vom Format 3½ Zoll und kleiner fällt jedoch die Biegsamkeit des Datenträgers, die diesem den Namen gegeben hat, weg. Hier befinden sich die Magnetplatten in festen Plastikgehäusen. Disketten werden in Diskettenlaufwerken betrieben. Ihr Vorteil gegenüber Magnetbändern besteht in einem direkten Zugriff auf die gespeicherten Daten.


Bild 4. Eine aufgeschnittene 5¼-Zoll-Diskette

Diskettenarten
Es gibt grundsätzlich zwei Arten von Disketten: einseitige und doppelseitige. Um eine doppelseitige Diskette nutzen zu können, benötigen Sie auch ein entsprechendes Diskettenlaufwerk mit zwei Schreib-/Leseköpfen, wie zum Beispiel die Commodore 1571. Eine mit »SS« (»single sided«) gekennzeichnete Diskette ist nur auf einer Seite geprüft, eine mit »DS« (»double sided«) gekennzeichnete hingegen auf beiden Seiten. Für die Güte eine Diskette gibt es noch die Angabe der Datenschreibdichte: Hierfür stehen die Buchstabenkombinationen »1D« oder »SD« (für »single density«, einfache Dichte), »2D« oder »DD« (für »double density«, doppelte Dichte) und »HD« (für »high density«, vier- oder mehrfache Dichte). Ein weiteres Gütemerkmal ist die Spurdichte, welche 24, 48 oder 96 tpi (tracks per inch) betragen kann.

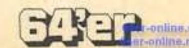
Diskettenmonitor
Ein Diskettenmonitor ist ein Programm, das es dem Anwender ermöglicht, Manipulationen direkt auf der Diskette vornehmen zu können.

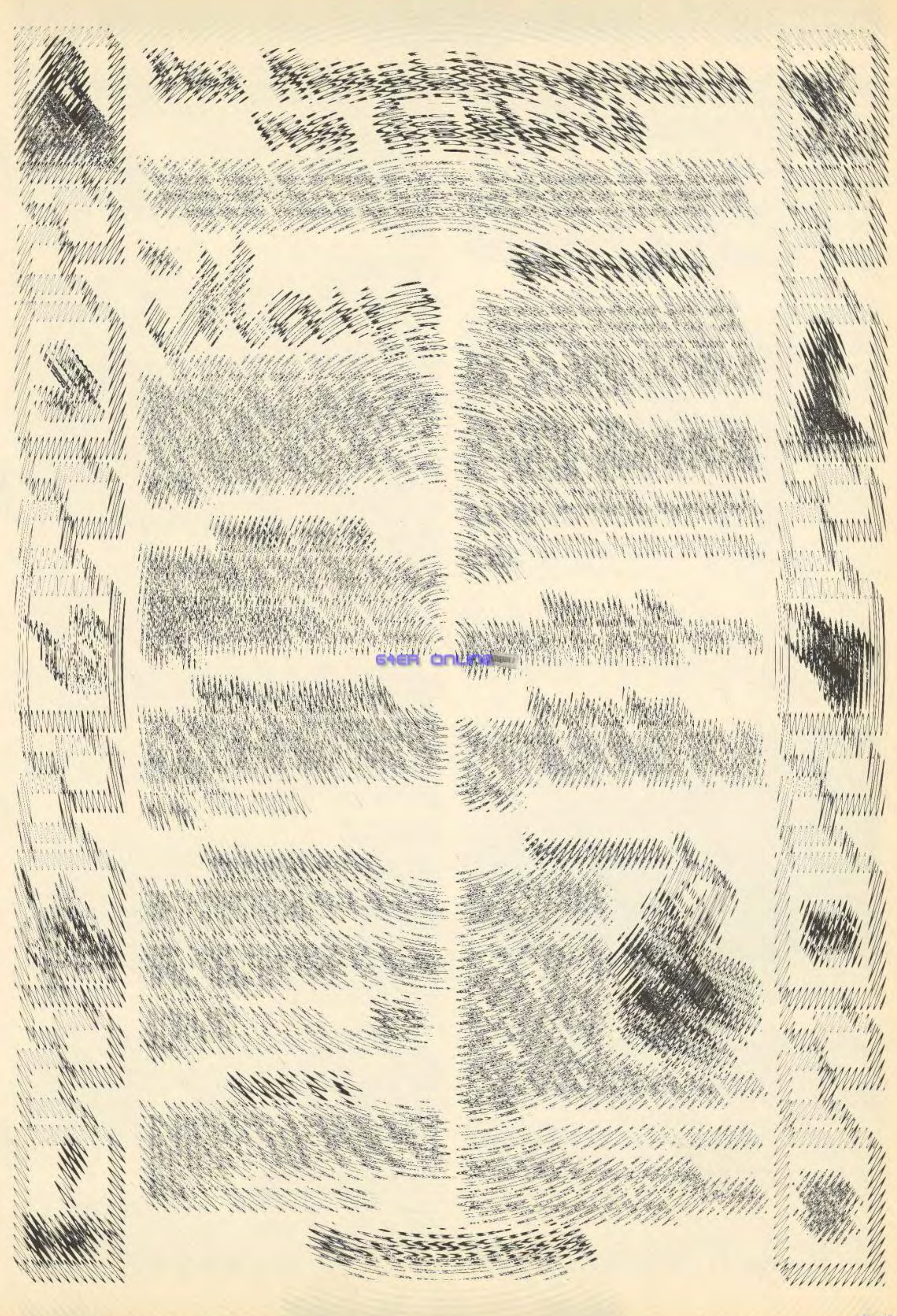
Das Prinzip ist folgendes: Dem Diskettenmonitor wird die Adresse des gewünschten Blocks angegeben, meist in hexadezimaler Form. Beispiel: Für den Block, in dem die BAM steht (Spur 18, Sektor 0) wäre dies $12 $00. Der Diskettenmonitor liest den gewünschten Block direkt von der Diskette in das RAM. Nun kann man den Block auflisten, anschauen, verändern und natürlich (in der geänderten Form) wieder auf die Diskette zurückschreiben lassen. Durch Änderung der entsprechenden Bytes in der BAM oder dem Directory können zum Beispiel Dateien oder ganze Disketten vor Löschen oder Überschreiben geschützt werden.

Diskettenorganisation
Um Daten auf einer Diskette zu lesen oder zu schreiben, muß das DOS die physikalische Position des Schreib-/Lesekopfes auf der Diskette erkennen. Die zur Positionsbestimmung erforderlichen Markierungen werden vom DOS auf die Diskette geschrieben, den Vorgang des Markensetzens nennt man »Formatieren«.

Beim 1541-Laufwerk von Commodore wird die Diskette beim Formatieren in 35 konzentrische Spuren (engl. Tracks) aufgeteilt. Jede Spur ist wiederum in kleinere Einheiten, sogenannte Sektoren (engl. Sectors), unterteilt. Je nach Position (außen oder innen auf der Diskette) hat eine Spur zwischen 17 und 21 Sektoren, von denen jeder 254 Datenbytes aufnehmen kann. Bei Commodore werden die Sektoren auch als Blöcke (Blocks) bezeichnet.

DOS (disk operating system)
Damit ein Computer überhaupt einen Massenspeicher ansprechen kann, muß er über ein spezielles Betriebssystem verfügen. Das DOS ist das »Diskettenbetriebssystem«





eines Computers. Bei vielen Heimcomputern sind Diskettenlaufwerke Erweiterungsbausteine, die extra gekauft werden müssen, wobei das DOS meist mitgeliefert wird.

Die Commodore-Laufwerke nehmen eine Sonderstellung ein, denn sie sind bereits ab Werk »intelligent«, das heißt, sie verfügen über ein eigenes, fest eingebautes Betriebssystem, das vom normalen Basic aus angesprochen werden kann. Diese Methode bietet den Vorteil, daß keinerlei Computerspeicher belegt wird. Außerdem kann man das Laufwerk gleichzeitig, aber unabhängig vom Computer, Diskettenmanipulationen vornehmen lassen (Beispiel: Validieren einer Diskette).

Duplex

Wenn auf einer Datenübertragungsleitung Daten nur in einer Richtung übertragen werden, bezeichnet man dies als Simplex-Betrieb. Dieses Verfahren wird wegen der fehlenden Rückmeldemöglichkeit (zum Beispiel zur Fehlerkorrektur) nur sehr selten angewandt.

Im Halbduplexbetrieb wird immer nur in eine Richtung übertragen. In der Hobby-DFÜ versteht man unter Halbduplex zumeist Vollduplexbetrieb ohne Echo.

Beim Vollduplexbetrieb ist die gleichzeitige Datenübertragung in beide Richtungen möglich, das heißt, beide Computer können gleichzeitig senden und empfangen. Im Echo-Betrieb wiederholt der empfangende Computer die Daten und sendet sie an den ersten Computer zurück.

Die Kommunikation zwischen Anwender und Mailbox findet nahezu ausschließlich im Vollduplexbetrieb statt.

Editor

Ein Programm, welches das Editieren von Text- oder Programmdateien ermöglicht. Unter dem Begriff »editieren« versteht man Modifikationen einer Datei, zum Beispiel Einfügen und Umstellen von Textbereichen, Verschiebungen und Löschungen.

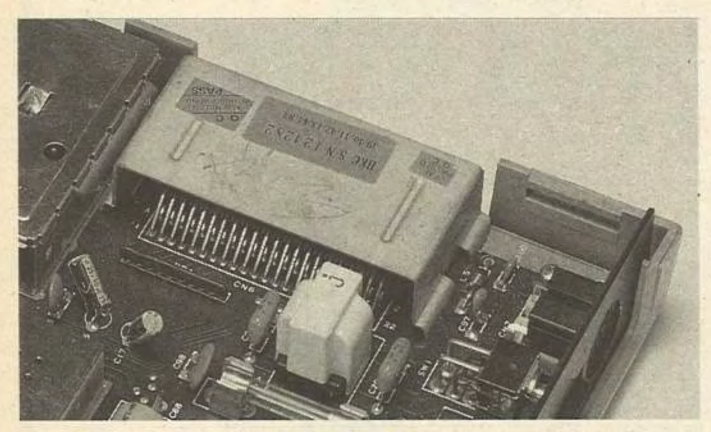


Bild 5. Der Expansions-Port des C64

Expansion-Port (Erweiterungs-Steckplatz)

An diesem Anschluß (Bild 5) des C64/C128 ist fast der gesamte Bus des Computers herausgeführt. Das bedeutet, daß alle zum Betrieb des Computers wichtigen Leitungen und Signale von außen zugänglich sind.

Damit läßt sich sehr viel anfangen, so können sehr gut Erweiterungsmodule von außen an den Computer angeschlossen werden. Ebenfalls kann hier ohne Aufschrauben des Computers ein anderes Betriebssystem angeschlossen werden.

Feld (engl. array)

Ein (mehrdimensionales) Datenfeld, auf dessen einzelne Elemente zugegriffen werden kann. Ein Schachbrett entspräche einem zweidimensionalen Feld von acht mal acht Elementen. Jedes einzelne Kästchen stellt ein Element dar, und jedes Element hat einen eigenen Wert.

Flußdiagramm (engl. flowchart)

Schematische Darstellung eines sequentiellen Programmablaufs unter Verwendung standardisierter Zeichen und Symbole, die Ereignisse oder Beziehungen zwischen Ereignissen während des Programmablaufs darstellen können.

Fremdformate

Ein großer Vorteil von CP/M ist die Softwarekompatibilität der CP/M-Computer untereinander. Durch unterschiedliche Aufzeichnungsformate können trotzdem nicht einfach Disketten ausgetauscht werden, denn fast jeder CP/M-Computer teilt sich die Diskette beim Formatieren systemspezifisch ein. Es gibt jedoch Laufwerke, die Formate von anderen Computer (eben jene Fremdformate) lesen und schreiben können. Dazu zählt auch die 1570/1571 von Commodore, welche bereits ab Werk in der Lage ist, Disketten folgender CP/M-Computer zu verwenden: Kaypro II, Osborne, Epson QX10 und Personal Computer unter CP/M 86. Mit Hilfe entsprechender Software kann sogar fast jedes CP/M-Format genutzt werden, was die Dateiübertragung erleichtert und Platz sparen hilft: Eine Diskette im Kaypro IV-Format faßt immerhin knapp 400 KByte - auch mit der Floppy 1571.

Hardcopy

Ausdruck von Texten oder Grafiken. Viele Computer verfügen über eine spezielle Hardcopy-Funktion, mit der der aktuelle Bildschirminhalt auf den Drucker ausgegeben wird.

Beim C64 gibt es entsprechende Programme (Hardcopy-Routinen), die geladen werden können und bei Aufruf den Grafik- oder Textbildschirm auf das Papier bringen.

Hardware

Überbegriff für alle mechanischen oder elektronischen Teile eines Computersystems, also alle materiellen Teile, sozusagen »was man anfassen kann«.

Hexadezimal

Das hexadezimale Zahlensystem beschreibt die Darstellung von Zahlen zur Basis 16. Die Dezimalzahlen von 0 bis 15 werden durch die Ziffern von 0 bis 9 und die Buchstaben A bis F (für 10 bis 15) dargestellt. Dieses System wird häufig zur platzsparenden Darstellung von Binärzahlen benutzt. Ein Byte läßt sich auf diese Weise mit nur zwei Symbolen eindeutig darstellen: Binär »11111011« entspricht Hex »FB«. Hexadezimalzahlen werden in der Regel durch ein vorangestelltes »$« gekennzeichnet.

Interpreter

Der Interpreter durchläuft für jeden Befehl einer bestimmten Programmiersprache eine festgelegte Befehlssequenz in Maschinensprache. Diesen Prozeß nennt man »interpretieren«. Beispiel: Wird ein Basic-Programm abgearbeitet und stößt der Basic-Interpreter auf den Befehl PRINT, so durchläuft er die entsprechende Bildschirmausgabe-Routine.

KI, Künstliche Intelligenz

Die zentrale Frage der KI ist: »Wie kann alles, was Menschen wissen, im Computer dargestellt und verarbeitet werden?«

Die Probleme der künstlichen Intelligenz können mit den bisherigen Programmiermethoden nicht mehr gelöst wer-



den; man braucht geeignetere Methoden, um Dinge der realen Welt (Gesetzmäßigkeiten, Zusammenhänge) auf einem Computer darzustellen.

Bekannte KI-Sprachen sind Lisp und Prolog. Diese unterscheiden sich grundlegend von Sprachen wie Basic oder Pascal.

Mailbox

Hinter dem Begriff Mailbox verbirgt sich nichts anderes als ein Computer, der mit dem Telefonnetz verbunden ist und mittels entsprechender Software Anrufe entgegennimmt und den Benutzer auf die Datenbestände zugreifen läßt.

Prinzipiell ist eine Mailbox ein »elektronischer Briefkasten«, in dem Daten (also auch Texte) abgelegt und wieder hervorgeholt werden können. Eine Mailbox umfaßt in der Regel aber wesentlich mehr Funktionen, zum Beispiel »Pinwände«, an die jeder seine Texte elektronisch »anheften« kann, so daß sie von weiteren Benutzern gelesen werden können.

Es gibt, besonders in den USA, auch (meist kommerzielle) Mailboxen, die mit Großrechnern und mehreren Telefonanschlüssen arbeiten, so daß sich auch mehrere Benutzer gleichzeitig im System befinden können und sogar Dialog- und Konferenzschaltungen möglich sind.

Modem

Ein Modem ist ein Gerät zur Datenübertragung, welches ohne Umwege an das Telefonnetz angeschlossen wird, also direkt (galvanisch) an die Telefonbuchse in der Wand. Bei einem Modem handelt es sich um ein Kombinationsgerät aus MOdulator und DEModulator, welches digitale Signale (vom Computer) in analoge (Ton-) Signale umwandelt. Somit können zur Tonübertragung geeignete Medien, wie zum Beispiel Telefonleitungen, für die Übermittlung digitaler Informationen eingesetzt werden. Der Anwender merkt hiervon wenig, denn die vom Modem erzeugten Signale werden direkt in das Telefonnetz eingespeist. Bei Bildschirmtext wird ein Modem zwischen Telefonanschluß und Btx-Endgerät geschaltet. Durch die galvanische Kopplung werden im Gegensatz zum Akustikkoppler Störungen durch Umweltgeräusche völlig ausgeschaltet.

Monitor

Bildschirmeinheit eines Computersystems (Bild 6). Vom Prinzip her ist ein Monitor nichts anderes als ein Fernseher, aber mit einem besseren, schärferen Bild und ohne eingebautes Empfangsteil (Tuner).

Man unterscheidet zwischen monochromen (einfarbigen) und Farbmonitoren.

Ebenfalls als Monitor werden Maschinensprache-Hilfsprogramme bezeichnet.

Programmiersprache, höhere
(engl. higher level language)

Alle Programmiersprachen, die sich durch »echte« Worte programmieren lassen (im Gegensatz zu Assembler), und bei denen die Anweisungen nicht im Befehlsformat des Prozessors formuliert sind, werden als höhere Programmiersprachen bezeichnet. Hier wird grundsätzlich noch zwischen Compiler- und Interpretersprachen unterschieden.

Prozessor

Eine Funktionseinheit innerhalb eines Computersystems, die nach vorgegebenen Anweisungen die Funktion anderer Einheiten steuern kann.

Die umfangreichsten Tätigkeiten dieser Art übernimmt die CPU (central processing unit), welche wegen ihrer zentralen Stellung im System auch Zentralprozessor genannt wird.

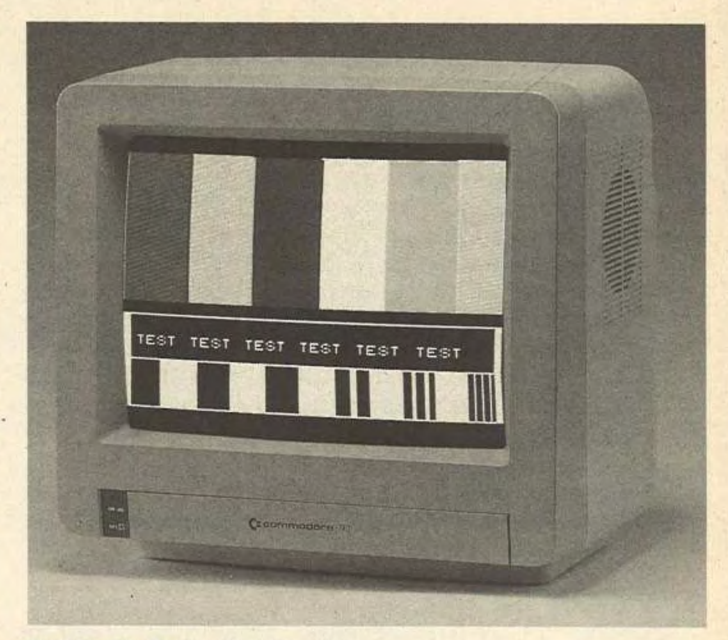


Bild 6. Ein Computer-Monitor

Daneben kann ein Computer aber noch mehrere spezialisierte Prozessoren (Coprozessoren) zur Steuerung peripherer Geräte oder zur effektiven Bearbeitung arithmetischer Problemstellungen (Arithmetikprozessor) aufweisen.

RAM (engl. random access memory)

Halbleiterspeicher, der sowohl gelesen als auch beschrieben werden kann. Wird die Betriebsspannung abgeschaltet (Ausschalten des Computers), so geht der Inhalt des RAMs verloren.

Reinigungsdiskette

Um eine Beschädigung von Diskette und/oder Laufwerk zu vermeiden, sollte der Schreib-/Lesekopf regelmäßig gereinigt werden. Zu diesem Zweck gibt es Reinigungsdisketten, bei denen trockene, halbtrockene, feuchte und Chromdioxid-Typen unterschieden werden. Vor trockenen Reinigungsdisketten warnen wir ausdrücklich, da diese den Dreck regelrecht abschmirgeln, was mehr Schaden anrichtet als nutzt. Beim halbtrockenen Verfahren muß vor dem Einlegen eine Flüssigkeit aufgebracht werden, so daß der Schmutz schonend gelöst wird. Feuchte Reinigungsdisketten enthalten auswechselbare Reinigungsvliese und werden immer nur einmal verwendet. Das letzte Reinigungsverfahren arbeitet mit einer chromdioxidbeschichteten Plastikfolie. Auch hier wird der Schmutz trocken abgerieben, was jedoch laut Hersteller den Kopf nicht angreift.

Register

Speichereinheit im Prozessor, in der Daten kurzzeitig während arithmetischer, logischer oder Übertragungsoperationen zwischengespeichert werden.

ROM, PROM, EPROM, EEPROM

Ein ROM (read only memory) ist ein Halbleiterspeicher, dessen Daten unveränderbar sind und nur ausgelesen werden können. Alle ROMs sind nichtflüchtig, der Speicherinhalt bleibt also nach Abschalten der Betriebsspannung erhalten. ROMs sind vor allem zur Speicherung eines Computer-Grundbetriebssystems im Einsatz, wobei die Daten und Steuerprogramme meist herstellerseitig in das ROM »hineingebrannt« werden.

Um die Kosten für die teure Herstellung eines zur Entwicklung notwendigen Mikrofilms zu sparen, verwendet man bei Kleinserien die nur einmal beschreibbaren (und nicht löschbaren) PROMs (programmable ROM).

Ein EPROM (erasable programmable ROM) ist im Gegensatz dazu ein mehrfach beschreibbarer Speicherbaustein. Zum Programmieren eines EPROMs benötigt man einen sogenannten EPROMer oder EPROM-Brenner, mit dem die Daten in das EPROM »gebrannt« werden. Das Löschen funktioniert durch Bestrahlung mit UV-Licht, wozu es spezielle EPROM-Löschgeräte auf dem Markt gibt.

Im Gegensatz dazu kann ein EEPROM (electrically erasable programmable ROM) auf elektrischem Wege gelöscht werden, was wesentlich schneller und vor allem schonender ist.

Schleife (engl. loop)
Teil eines Computerprogramms, der solange wiederholt wird, bis eine bestimmte Bedingung erfüllt ist.

Schnittstelle
Allgemein die Verbindungsstelle zweier miteinander in Verbindung stehender Systeme. Es hat sich die Unterscheidung zwischen Mensch-Maschine-Schnittstelle und Maschine-Maschine-Schnittstelle eingebürgert.

Die Mensch-Maschine-Schnittstelle (auch Benutzerschnittstelle genannt) umfaßt die Bedienelemente eines Computers, zum Beispiel Tastatur oder Bildschirm, aber auch denjenigen Teil der Software, der mit dem Benutzer in Dialog tritt (bei der Software spricht man hier von der »Benutzeroberfläche«).

Über eine Maschine-Maschine-Schnittstelle erfolgt der Austausch von Daten oder Steuerinformationen zu den peripheren Geräten (Drucker, Diskettenlaufwerk).

Schreibschutz
Um Daten vor einem versehentlichen Löschen oder Überschreiben zu schützen, verfügen die meisten Datenträger über eine Schreibschutzvorrichtung. Bei einer Tonbandkassette wird dazu ein kleines Plastikteil aus dem Gehäuse herausgebrochen. Die Aufnahmetaste eines Kassettenrecorders kann nun nicht mehr gedrückt werden, wenn sich diese Kassette im Laufwerk befindet. Ähnlich funktioniert das Ganze bei einer 5¼-Zoll-Diskette: Hier ist an der Seite eine sogenannte »Schreibschutzkerbe« vorhanden. Wird diese Öffnung mit einem Stück undurchsichtigen Klebeband verschlossen, kann diese Diskettenseite nur noch gelesen werden. Ein Überschreiben der Daten oder gar Formatieren der Diskette wird somit verhindert. Bei der 3½-Zoll-Diskette befindet sich ein kleiner Schiebeschalter im Diskettengehäuse, der den Schreibschutz ein- und ausschaltet.

Software
Die Software ist der nicht-materielle Bestandteil eines Computersystems, also die Programme und Daten.

Sprites
Vom Anwender freidefinierbares, grafisches Objekt. Auf dem C64 ist ein Sprite 21 x 24 Punkte groß. Sprites können dort sowohl farbig als auch schwarzweiß dargestellt werden. Der Vorteil dieser Objekte besteht darin, daß sie unabhängig vom Bildschirminhalt als eigenständige Fläche bewegt werden können. Ohne programmtechnische Tricks können auf dem C64 bis zu acht dieser Gebilde dargestellt und bewegt werden.

Sprites werden hauptsächlich benutzt, um bei Computerspielen Figuren unabhängig vom gerade angezeigten Bildschirm bewegen zu können.

Systemsoftware
Hierzu zählen alle Systemprogramme, die nicht anwendungsbezogen, sondern für den Betrieb des Computers grundsätzlich erforderlich sind. Dazu gehört auch das Betriebssystem eines Computers.

Übertragungsprotokoll
Bei der Kommunikation von Computern im Datenaustausch befolgen die beteiligten Geräte einen Standard, der die Einzelheiten des Informationsaustausches regelt. Dies ist sowohl bei der Kommunikation von Computern mit Computern als auch bei der Datenübertragung eines Computers zu einem Peripheriegerät (wie zum Beispiel einem Drucker) notwendig. Der gemeinsame Standard wird in Anlehnung an die Sprechweise der Diplomaten Protokoll genannt.

Im Quittungsbetrieb (engl. handshaking) teilen sich sendender und empfangender Computer gegenseitig mit, wann sie zum Empfang oder zur Übertragung weiterer Informationen bereit sind. Würde auf das Protokoll verzichtet oder der Austausch von Quittungen (englisch: acknowledge) unterbleiben, so gingen mit ziemlicher Sicherheit Informationen verloren.

User-Port (Schnittstelle für den Benutzer)
Dieser Port des C64/C128 (Bild 7) ist ein variabler Ein-/Ausgabe-Anschluß. Variabel bedeutet in diesem Zusammenhang, daß man jede der acht Datenleitungen, die hier herausgeführt sind, unabhängig auf Ein- oder Ausgang schalten kann.

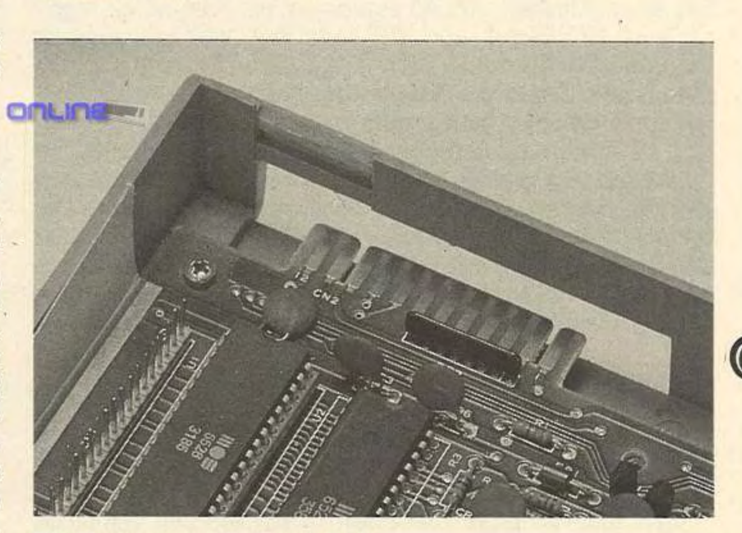


Bild 7. Der User-Port des C64

So lassen sich hier zum Beispiel Meßwerte aufnehmen, die dann von einem Programm weiterverarbeitet werden können. Auch ist es möglich, über den User-Port diverse Geräte (Lichtorgel, Alarmanlage) zu steuern.

Der User-Port findet aber hauptsächlich Anschluß beim Betrieb von Druckern mit Centronics-Schnittstelle und Akustikkopplern.

Variable
Symbolischer Name, an den ein Wert oder eine Zeichenkette gebunden ist. Indem man sich (in einem Programm) auf den Namen bezieht, bezieht man sich auf den in der Variable enthaltenen Wert oder die darin enthaltene Zeichenkette.

Es gibt verschiedene Variablentypen, die unterschiedliche Arten von Werten beinhalten können. So ist eine Integervariable ausschließlich zur Aufnahme ganzer Zahlen (zum Beispiel 3, 165 oder -91) bestimmt. (pd)





Basic-Kurs von Anfang an

Illustration: Rolf Boyke

Basic-Kenntnisse gehören heute zur Allgemeinbildung! Ein Commodore-Besitzer muß einfach Basic können, um mit seinem Computer vernünftig zu arbeiten. Denn die meisten Programme sind in Basic geschrieben. Sie werden staunen, wie schnell Sie ein funktionierendes Basic-Programm geschrieben haben! Hier zeigen wir allen Neulingen, wie man programmiert.

Alle Anfänge werden begleitet von guten Vorsätzen! Am Anfang dieses Basic-Kurses möchte ich keine Ausnahme machen und sogar soweit gehen, Ihnen meine guten Vorsätze offen darzulegen.

1. Ich habe den Kurs so geschrieben, daß Sie nur wenige Voraussetzungen mitbringen müssen, um ihn zu verstehen. Ich gehe davon aus, daß Sie die ersten Kapitel des Handbuches gelesen und angewendet haben, welches Ihnen beim Kauf des Computers mitgeliefert worden ist.
2. Ich möchte gern ohne Fach-Chinesisch erklären. Wo ich ohne Fachausdrücke nicht auskommen kann - und das wird natürlich öfters vorkommen - werde ich sie erklären.
3. Der Kurs beginnt ganz langsam, mit vielen leichten Beispielen und Wiederholungen.
4. Der Kurs soll aber nicht langweilig sein. Er wird daher bald an Tempo zunehmen, im Vertrauen darauf, daß Sie sehr bald keine reinen Anfänger mehr sein werden.
5. Ich gehe davon aus, daß Sie sich nicht ein rein theoretisches Basic aneignen, sondern programmieren lernen wollen. Das bedeutet aber, daß ich Ihnen auch Bedienungshinweise und Erklärungen der Arbeitsweise des Computers geben muß - so wenig wie möglich und so viel wie nötig.
6. Der letzte Vorsatz ist eigentlich eine Aufforderung an Sie, liebe Leser. Wenn Sie etwas nicht verstanden oder spezielle Fragen haben, dann schreiben Sie mir bitte über den Verlag. Ich werde die Fragen entweder direkt beantworten oder aber die Fragen sammeln und Ihnen im 64'er eine eigene Frage-Antwort-Folge widmen.

Bevor wir anfangen, will ich noch schnell die Methode erwähnen, mit der Sie am besten den Kurs verfolgen und den Stoff nachvollziehen können.

Ich möchte diese Vorgehensweise »SIMULTAN-METHODE« (Bild 1) nennen, was nichts anderes bedeutet, als das Heft mit Basic-Kurs vor dem Computer sitzend zu lesen und alle Angaben, die mit dem Zeichen »⟶« gekennzeichnet sind, sofort einzutippen und auszuprobieren.

Ich bin mir bewußt, daß ich dadurch eine Arbeitsweise begünstige, die von vielen Programmierspezialisten generell verteufelt und ganz besonders der Sprache Basic vorgeworfen wird, nämlich »Hackerei«.

Aber gerade für Anfänger ist die Eigenschaft von Basic, die kleinsten Programmteile sofort ausprobieren zu können, von großem Vorteil.

So, jetzt habe ich keinen Grund mehr, weitere schöne Einleitungsphrasen zu dreschen. Jetzt soll's losgehen.

Lektion 1: Wie bediene ich den Computer?

Voraussetzung für die Arbeit mit einem Computer ist eine Einrichtung, dem Computer mitzuteilen, was er tun soll und eine andere Einrichtung, um zu sehen, was er tut beziehungsweise was er getan hat. Dazu dient als allererstes die Tastatur und der Bildschirm.

Sobald Sie den Computer einschalten, ist das Eingabegerät Tastatur direkt mit dem Ausgabegerät Bildschirm verbunden. Im Handbuch Ihres Computers haben Sie sicher schon nachgelesen, wie die Tastatur funktioniert und wie damit Zeichen auf den Bildschirm gezaubert werden. Ich will das nicht im Detail wiederholen, aber ein paar Gedanken ist das Thema »Tastatur« schon wert.

Ich teile die Tasten in drei verschiedene Typen ein:
- Funktionstasten
- Zeichentasten
- Steuertasten

Mit *Funktionstasten* werden die 4 senkrecht untereinander angeordneten Tasten <F1> bis <F7> rechts außen bezeichnet. Sie sind universell für alle möglichen Funktionen einsetzbar, können diese Funktionen aber erst dann erfüllen, wenn sie entsprechend programmiert worden sind. Aus diesem Grund werde ich sie erst später behandeln und ich lasse sie vorläufig links liegen.

Zeichentasten erzeugen, wie ihr Name sagt, Zeichen. Dazu gehören Buchstaben und Zahlen, mathematische Zeichen (Addition, Multiplikation, runde Klammern etc.), Symbole (Dollar, Anführungsstrich, Pfeile und so weiter) und grafische Zeichen. Die grafischen Zeichen stehen nicht oben auf den Tasten, sondern auf ihrer Vorderseite.

Jede Zeichentaste bietet mehrere Zeichen und Symbole zur Auswahl. Um die verschiedenen Zeichen einer Taste auszuwählen und auf den Bildschirm zu bringen, verwenden wir spezielle Tasten wie <SHIFT>, <CTRL> und so weiter. Diese Tasten nenne ich *Steuertasten*.

Sie steuern auf sehr direkte Art und Weise alle Vorgänge auf dem Bildschirm. Aber sie erlauben auch, innerhalb von Programmen deren Abläufe zu steuern und zu verändern.

Ich halte diese Tasten für wichtig genug, um trotz ihrer Behandlung im Handbuch von Commodore etwas näher auf sie einzugehen.

Steuertasten sind also die CTRL-Taste, mit der die acht verschiedenen Zeichenfarben schwarz bis gelb eingestellt werden können, aber auch die SHIFT-Taste, welche den rechten Teil der Zeichen und grafischen Symbole umschaltet. Dazu gehört auch die Taste links unten, die das Firmenzeichen von Commodore trägt und welche die linken Symbole umschaltet.

Dazu gehören schließlich und letztlich die Cursor-Steuertasten <CRSR>, die INST/DEL-Taste und die CLR/HOME-Taste.

Diese letztgenannten Tasten sind eng mit der Wirkungsweise des sogenannten Editors verbunden.

Editor ist ein englisches Wort und heißt soviel wie Redakteur. In unserem Fall ist der Editor ein im Computer fest eingebautes Programm, welches wie ein Redakteur dafür sorgt, daß auf dem Bildschirm alle Zeichen und Symbole in den entsprechenden Farben auf den richtigen Platz kommen.



Illustration: Rolf Boyke

Bild 1. »Learning by doing«

Ein sichtbares Hilfsmittel, dem Editor unsere Wünsche und Vorstellungen mitzuteilen, ist der *Cursor*.

Die Bedeutung und seine Handhabung haben Sie ja sicher schon dem Handbuch entnommen. Das Wort »Cursor« kommt nicht vom englischen »curse« = verfluchen (obwohl das Verhalten des Cursors oft dazu verleitet), sondern aus dem Lateinischen, wo es »Läufer« bedeutet.

Der Cursor läuft also über den Bildschirm, gesteuert vom Editor. Sie können seinen Lauf auch steuern, und zwar mit den beiden Cursor-Steuertasten.

Eine andere Steuerung des Cursors bietet die CLR/HOME-Taste rechts oben. Allein gedrückt – dann gilt <HOME> – bringt sie den Cursor »nach Hause« nämlich in die linke obere Ecke des Bildschirms. Wird die Taste mit <SHIFT> umgeschaltet, bedeutet sie <CLR> – das heißt <CLEAR>, auf deutsch Freimachen oder Löschen. Jetzt wird nämlich der Cursor nach links oben gebracht und zusätzlich der Bildschirm gelöscht.

Probieren Sie bitte diese Cursor-Bewegungen aus.

→ Am besten fahren Sie den Cursor in die Mitte des Bildschirms, tippen ein paar beliebige Buchstaben ein und drücken dann die HOME-Taste.

→ Wiederholen Sie das Ganze und verwenden Sie dann die CLR-Taste.

Der Editor bietet uns noch einen anderen sehr lobenswerten Service. Er berücksichtigt nämlich, daß wir alle sehr menschliche Menschen sind, die nicht nur viele Fehler machen, sondern auch immer wieder ihre Meinung ändern. Der Editor erlaubt uns, Fehler zu korrigieren oder bereits getippte Zeichenfolgen abzuändern.

→ Sie können mit dem Editor ungestraft über vorhandene Zeichen fahren, ihn nach Lust und Laune anhalten und dort, wo er gerade blinkt, ein neues Zeichen eintippen.

Ich nenne das »überschreiben«. Wo wir ein Überschreiben nicht anwenden können, hilft uns die INST/DEL-Taste (rechts oben) weiter.

<INST> ist die Abkürzung für Insert, das heißt soviel wie einfügen.

<DEL> bedeutet Delete, und das heißt entfernen oder auslöschen.

Im Handbuch steht nur eine kurze Beschreibung der Wirkung dieser Taste, die aber nicht unbedingt ausreichend ist. Schon erste Versuche zeigen, daß die Taste ihre Tücken hat. Ich bin dafür, DEL und INST einfach auszuprobieren.

→ Schalten Sie den Computer aus und dann wieder ein.

→ Fahren Sie mit dem Cursor auf das A von READY. Er steht jetzt an der dritten Stelle vom linken Bildrand.

→ Drücken Sie die DEL-Taste. Das E ist verschwunden, es steht nur noch RADY da, der Cursor blinkt immer noch auf dem A, er ist aber jetzt an die zweite Stelle vom linken Bildrand gerückt.

→ Ein zweiter Druck auf die DEL-Taste reduziert das Wort auf ADY, und der Cursor steht jetzt auf der ersten Stelle.

Die DEL-Taste löscht also das links neben dem Cursor stehende Zeichen und verschiebt den Cursor mitsamt dem ganzen rechten Schwanz eine Stelle nach links. Was passiert aber, wenn der Cursor am linken Bildrand angestoßen ist?

→ Ein dritter Druck auf die DEL-Taste bringt den Cursor an den rechten Bildrand eine Zeile darüber, nur da gibt es gerade nichts zu löschen. Seinen ADY-Anhang läßt er am Anfang der unteren Zeile zurück.

→ Lassen Sie die DEL-Taste gedrückt. Der Cursor läuft kontinuierlich weiter nach links, löscht alles, was ihm in den Weg kommt – solange, bis er »zu Hause« links oben angekommen ist.

Der geSHIFTete Teil dieser Taste, nämlich <INST> ist ebenso trickreich.

→ Schalten Sie den Computer aus und wieder ein.

→ Fahren Sie mit dem Cursor wieder auf das A von ADY.

→ Drücken Sie die INST-Taste (SHIFT + INST/DEL). Der Cursor bleibt auf seiner dritten Stelle vom linken Bildrand stehen. Aber das Zeichen, auf dem er blinkt und alles, was rechts von ihm steht, wird eine Stelle nach rechts geschoben. Wo der Cursor blinkt, entsteht eine freie Stelle, auf die direkt ein neues Zeichen geschrieben werden kann.

→ Durch mehrfaches Drücken der INST-Taste werden mehrere Stellen freigeschoben, in die mehrere Zeichen hintereinander eingefügt werden können.



Dieses Freimachen beziehungsweise Einfügen geht nicht in beliebiger Länge, da der Editor gewissen Einschränkungen unterliegt. Ich würde natürlich am liebsten jetzt weiter über den Editor mit allen seinen Regeln und Einschränkungen dozieren. Und irgendwann muß ich es auch noch tun, denn beim Experimentieren werden Sie sicher noch darauf stoßen und wissen dann vielleicht nicht weiter.

Damit Sie aber nicht ungeduldig werden, möchte ich auf weitere Feinheiten vorläufig verzichten und das bisher Gelernte für die ersten Schritte in Basic anwenden.

FAZIT
Die Zeichen- und Steuertasten erzeugen eine direkte Wirkung auf dem Bildschirm. Der Bildschirm wird von einem eingebauten Programm verwaltet - dem Editor. Er ermöglicht beliebige Änderungen und Korrekturen von bereits eingegebenen Texten und Zahlen.

Lektion 2: Computer bedeutet Rechenmaschine

Diese Überschrift weist uns den Weg für das weitere Vorgehen.

Wenn jeder simple Taschencomputer rechnen kann, dann müssen es die Commodore-Computer auch können. Dazu will ich Ihnen noch schnell die Symbole der mathematischen Funktionen aufschreiben, die der Computer verwendet.

FUNKTION	SYMBOL	BEISPIEL
Addition	+	3+2
Subtraktion	-	3-2
Multiplikation	*	3*2
Division	/	3/2
Potenz	↑	3↑2

Die unterste Funktion ist wahrscheinlich die für Sie ungewohnteste. Das Beispiel wird im Klartext als »drei hoch zwei« oder als »drei Quadrat« gelesen.

Beim Taschenrechner wird normalerweise eine Rechnung so eingegeben:

⟶ 3 + 2 =

Aber beim C64 rührt sich da gar nichts, außer daß diese Zeile auf dem Bildschirm steht.

Und warum passiert nichts?

Überlegen Sie - alles, was wir gemacht haben - war, per Tastendruck Zahlen und Symbole einzutippen. Wir haben lediglich den Editor auf Trab gehalten, den Computer selbst haben wir damit nicht aufgeweckt.

Dieser Faulpelz schläft nämlich so lange, bis er einen Auftrag bekommt, etwas auszuführen. Für diesen Tritt, der ihn hinter dem Ofen hervorscheucht, brauchen wir eine der vorher zwar erwähnten, aber noch nicht eingesetzten Steuertasten zur Programmsteuerung.

Sie sitzt am rechten Rand der Tastatur und ist mit <RETURN> gekennzeichnet.

Schreiben wir also die Zeile von oben noch einmal. Dieses Mal bitte ich Sie aber, nicht die Eingabe-Technik der Taschencomputer zu verwenden, sondern mir eine »lesbare« Schreibweise nachzumachen - Sie werden gleich sehen, warum.

Schließen Sie die Eingabe mit der RETURN-Taste ab. Das sieht dann so aus:

⟶ SUMME = 3 + 2
⟶ Drücken Sie die RETURN-Taste

Aber wir sehen ja noch immer nichts!

Aller Anfang ist schwer. Wir müssen berücksichtigen, daß der Computer nichts, aber auch gar nichts von allein macht.

Für alles braucht er eine Anweisung. Nun, oben haben wir ihm die Anweisung gegeben, die Summe aus 3 plus 2 zu bilden. Das hat er auch gemacht, glauben Sie es mir. Aber wir haben ihm nicht gesagt, was er mit der Summe machen soll, und so hat er sie sich einfach nur gemerkt.

Wir wollen sie natürlich auf dem Bildschirm sehen. Dazu müssen wir dem Computer eine spezielle Anweisung geben.

Dieser Befehl, irgend etwas auf dem Bildschirm auszudrucken, lautet in Basic

PRINT.

Nach PRINT, was auf deutsch »drucken« heißt, geben wir dem Computer an, was er ausdrucken soll - in unserem Beispiel die Summe.

⟶ PRINT SUMME
⟶ Drücken Sie die RETURN-Taste.

Heureka! Endlich sind wir weitergekommen. Auf dem Bildschirm steht, vom Editor in die nächste Zeile gebracht, die Zahl, welche der Computer für die SUMME ausgerechnet hat.

Basic-Befehl Nr. 1 PRINT
druckt alles, was hinter dem Wort »PRINT« steht, auf dem Bildschirm aus.

Ich schlage vor, daß Sie mit dem ersten Wort der Sprache Basic, das Sie nun gelernt haben, noch ein bißchen üben.

Bilden Sie das Produkt von 15 mal 14. Das geht doch einfach, oder?

⟶ PRODUKT = 15 * 14
⟶ nicht vergessen <RETURN> zu drücken
(Sie wissen doch, mit RETURN geben wir dem Computer die Anweisung, die getippte Zeile auszuführen)
⟶ PRINT PRODUKT
⟶ <RETURN> drücken

Wenn Sie gut Kopfrechnen können, dürfen Sie das Resultat nachprüfen. Oder vielleicht nehmen Sie einen Taschenrechner!

Ich wette, irgend ein Leser sagt jetzt: »Was soll der Quatsch. Man kann doch die Rechnung 15*14 direkt mit PRINT ausführen. Der Umweg über SUMME = oder PRODUKT = ist doch völlig unnötig«. Tja, lieber Leser, das stimmt in der Tat. Diese Kurzform ist erlaubt, und ich will Sie Ihnen gleich vorführen, vielleicht mit der Division. Wir wollen 3 durch 2 teilen. Geben Sie dazu ein:

⟶ PRINT 3/2
⟶ <RETURN> drücken

Wir erhalten sofort den Wert 1,5 - aber halt!! Er ist 1.5, also nicht mit Komma, sondern mit Dezimalpunkt geschrieben. Das ist die amerikanische Schreibweise, an die Sie sich gewöhnen müssen; als Preis dafür, daß Sie mit amerikanischen Computern arbeiten.

Also - der Befehl PRINT führt eine nachfolgende Rechnung direkt aus und druckt das Resultat auf den Bildschirm. Ist demnach meine oben genannte Methode, die Rechnung in zwei Schritten auszuführen, tatsächlich Quatsch?

Meine Antwort ist ein klares Nein. Beide Methoden haben ihre Berechtigung, und ich will Ihnen auch den Unterschied zeigen.

Mit der ersten Methode (wir haben sie bei SUMME und PRODUKT angewendet) erhält der Computer zuerst den Auftrag, eine Rechnung durchzuführen. Das Resultat merkt er sich unter dem Stichwort SUMME beziehungsweise PRODUKT. Haben Sie es gelesen? Er merkt sich das

Resultat und legt es in einem Speicher (Bild 2) ab. Das bedeutet, daß wir es mit PRINT so oft wir wollen, auf den Bildschirm drucken können. Geben Sie jetzt gleich nochmal ein:

→ PRINT SUMME
→ <RETURN> drücken

Und siehe da, das Resultat von vorhin erscheint wieder. Dasselbe geht mit dem Stichwort PRODUKT.

Sie brauchen übrigens die Zeile nicht neu einzugeben. Vergessen Sie den Service des Editors nicht!

→ Fahren Sie mit dem Cursor auf die letzte PRINT-Zeile und überschreiben Sie das Wort SUMME mit dem Wort PRODUKT, dann muß nur noch die RETURN-Taste gedrückt werden.

Und noch einmal siehe da, wir erhalten das Resultat der Multiplikation wieder.

So, und was ist mit der Division?

Leider, leider! Da geht nichts mehr. Wir haben kein Stichwort für das Resultat vorgegeben, und der Computer hat sich auch nichts gemerkt.

Das ist also der Unterschied:

• PRINT 3 + 2	• SUMME = 3 + 2 • PRINT SUMME
Das Resultat der Rechnung wird ausgedruckt	Das Resultat der Rechnung wird unter dem Stichwort SUMME gespeichert und dann ausgedruckt

Die linke Anwendung des Befehls PRINT ist einfach, und Sie können sie immer verwenden, um nach Art des Taschenrechners schnell einmal irgendwelche Rechnungen zu machen. Aufgezeichnet wird das Resultat allerdings nur auf dem Bildschirm. Im Computer selbst bleibt nichts gespeichert.

Die rechte Art ist schwieriger, aber interessanter. Es ist eigentlich erstaunlich, daß wir dem Computer die Anweisung geben, sich eine Zahl unter einem Namen oder – wie ich es vorhin genannt habe – unter einem Stichwort zu merken, ohne einen Basic-Befehl geben zu müssen.

Es gibt tatsächlich einen Befehl dafür, er heißt:

LET

Die Anweisung mit dem Stichwort »SUMME« sieht unter Zuhilfenahme des Befehls LET so aus:

LET SUMME = 3 + 2

In unseren Sprachgebrauch übersetzt heißt das ungefähr: »Laß die Summe gleich dem Resultat von 3 plus 2 sein«. In der Computersprache nennt man das eine ZUWEISUNG.

Wir können demnach beliebige Stichwörter hernehmen und ihnen mit dem LET-Befehl Zahlenwerte zuweisen, der Computer merkt sie sich alle.

Wie würden Sie zum Beispiel dem Stichwort A (kürzer kann es nicht sein) den Wert 375 zuweisen, dann dem Stichwort X den Wert 15 und schließlich die Division von vorhin, die der Computer sich nicht gemerkt hat, jetzt dauerhaft durchführen ?

Versuchen Sie es:

So muß es aussehen:

→ Löschen Sie den Bildschirm (<SHIFT>+<CLR/HOME>)
→ LET A = 375 (<RETURN> drücken)
→ LET X = 15 (<RETURN> drücken)
→ LET QUOTIENT = 3/2 (<RETURN> drücken)

Jetzt sind im Computer die zwei Werte unter den Stichworten A, X und das Ergebnis der Division unter dem Namen QUOTIENT gespeichert.

Mit PRINT-Befehlen können wir alle drei auf den Bildschirm bringen:

→ PRINT A (<RETURN> drücken)
→ PRINT X (<RETURN> drücken)
→ PRINT QUOTIENT (<RETURN> drücken)

SUMME, PRODUKT, A, X und QUOTIENT, die wir bisher »Stichwörter« oder »Namen« genannt und welchen wir mit dem LET-Befehl einen Wert zugewiesen haben, werden mit einem eigenen Fachausdruck bezeichnet.

Sie heißen Variable. Ich werde später noch mehr über sie berichten. Zunächst bitte ich Sie lediglich, sich den Namen zu merken – manchmal werde ich vielleicht auch noch ein Stichwort sagen.

Wozu brauchen wir den Befehl »LET«? Am Anfang haben wir ohne ihn den Stichwörtern Resultate zugewiesen!

In Basic kann in der Tat der Befehl LET weggelassen werden. Wichtig ist lediglich, daß zuerst die Variable dasteht, gefolgt von dem Gleichheitszeichen (=) und dahinter der Wert (oder der durch die Rechnung auszurechnende Wert).

Basic-Befehl Nr. 2 LET
weist einer Variablen einen Wert zu. Er kann auch weggelassen werden. Es genügt der Name der Variablen, ein Gleichheitszeichen und der zugewiesene Wert.

Interessant ist übrigens, daß der Computer sich ein und dasselbe Stichwort nur einmal merkt. Wenn Sie zuerst dem Stichwort QUOTIENT den Wert 3/2 zuweisen und gleich danach den Wert 15/3, erhalten Sie mit dem PRINT-Befehl nicht die Zahl 1.5, sondern 5.

Das erste Resultat hat der Computer in seinem Speicher durch das zweite überschrieben. Probieren Sie es aus:

→ PRINT QUOTIENT (<RETURN> drücken)
→ LET QUOTIENT = 15/3 (<RETURN> drücken)
→ PRINT QUOTIENT (<RETURN> drücken)

Zuerst kommt der alte Quotient von vorhin, danach der neue Wert.

FAZIT
1. Nicht nur wir können Zeichen auf den Bildschirm bringen – der Computer selbst kann das auch. Der Befehl PRINT druckt alles, was dahinter steht, auch Ergebnisse von Rechenanweisungen, auf den Bildschirm.
2. Unter einem Stichwort (Variable genannt) kann sich der Computer Zahlenwerte merken, die beliebig oft mit dem PRINT-Befehl ausgedruckt werden können. Voraussetzung ist lediglich, daß mit dem LET-Befehl dem Stichwort der Zahlenwert zugewiesen wird. Das Wort »LET« selbst kann auch weggelassen werden.

Lektion 3: PRINT – der Universalbefehl

3.1 Zahlenwerte ausdrucken

Bisher haben wir nur Zahlenwerte und zugehörige Variable mit PRINT auf dem Bildschirm ausgedruckt.

Ist Ihnen das viele PRINT-Tippen, so wie bei den Beispielen oben, auch so lästig wie mir?

Diesen Dauerversuch mit unzähligen PRINT-Befehlen können Sie vereinfachen. Alle Commodore-Computer sind nämlich Meister der Abkürzung. So drastisch, wie wir den Befehl LET durch Weglassen abgekürzt haben, dürfen wir den Befehl PRINT nicht behandeln. Aber immerhin: Er wird mit dem Fragezeichen abgekürzt.

Das Befehlswort »PRINT« kann durch das »?« ersetzt werden.

Statt PRINT 123
dürfen Sie schreiben ? 123

Machen Sie bitte Gebrauch von dieser Abkürzung, wann immer Sie wollen.

Ich werde allerdings stets das ganze Befehlswort verwenden, damit der Text gut lesbar bleibt.

Die Sprache Basic gewährt uns noch eine Marscherleichterung. Es ist nämlich erlaubt, mehrere auszudruckende Werte hinter einen einzigen PRINT-Befehl zu setzen.

Zuerst will ich Ihnen zeigen, wie Sie es nicht machen dürfen – probieren geht über studieren.

Haben Sie die alten Stichwörter noch im Computer? Wenn ja, dann brauchen Sie die folgenden Zuweisungen nicht nochmal eingeben. Ich wiederhole sie für alle Leser, die inzwischen ihren Computer ausgeschaltet und damit sein Gedächtnis gelöscht haben.

→ `A = 375` (<RETURN> drücken)
→ `X = 15` (<RETURN> drücken)
→ `QUOTIENT = 15/3` (<RETURN> drücken)

Sie sehen, ich habe LET weggelassen.

So, die Werte haben ihr Stichwort, unter dem sie der Computer in seinem Speicher wiederfindet. Jetzt wollen wir sie mit einem einzigen PRINT-Befehl ausdrucken:

→ `PRINT A X QUOTIENT` (<RETURN> drücken)

Der Computer druckt eine Null aus. Wie gesagt, so geht's nicht.

Wenn ich Ihnen sage, warum eine Null erscheint, werden Sie gleich auf die Lösung des Problems kommen. Was passiert da?

Für den Computer ist eine Leerstelle (zwischen den Variablen) genau so ein Zeichen wie jeder Buchstabe. Das heißt, daß er nach dem Drücken der RETURN-Taste in seinem Speicher nach einem Stichwort »A X QUOTIENT« sucht – und das findet er natürlich nicht.

Ihnen ist sicher klar, daß wir also die drei Stichwörter voneinander trennen müssen. Die Rechtschreibregeln von Basic erlauben keine Wörtertrennung mit Leerzeichen, sondern mit dem Semikolon »;«.

Verbessern Sie bitte (durch einfaches Einfügen von zwei Semikolons) die letzte Zeile oben:

→ `PRINT A;X;QUOTIENT` (<RETURN> drücken)

Jetzt stimmt's.

Bei diesem Beispiel kann ich Ihnen noch etwas zeigen, diesmal eine Eigenheit – oder besser gesagt – eine Vorsorge des guten alten Editors. Schauen Sie sich genau an, wie der Editor die drei Zahlen 375, 15 und 5 auf den Bildschirm geschrieben hat. Vor der ersten Zahl steht eine Leerstelle, zwischen den Zahlen sind zwei Leerstellen.

Das kommt daher, daß der Editor vor jeder Zahl eine Stelle reserviert, um im Fall einer negativen Zahl Platz für das Minuszeichen zu haben. Und zwischen den Zahlen hält er natürlich einen Abstand frei. Wie bereits gewohnt, wollen wir das am Computer überprüfen.

→ Bitte geben Sie die drei Werte unserer Variablen als negative Zahlen, also mit einem Minuszeichen ein. Wenn Sie die Zuweisungen noch auf dem Bildschirm stehen haben, geht es schneller mit Hilfe des Editors. Fahren Sie mit dem Cursor auf die jeweils vorderste Ziffer, schaffen Sie mit der INSERT-Taste (<SHIFT> + <INST/DEL>) eine Leerstelle und tippen Sie ein Minuszeichen ein, danach wie immer ein <RETURN>.
Fahren Sie mit dem Cursor auf die Zeile mit dem PRINT-Befehl und drücken Sie die RETURN-Taste.

Ja, der Editor macht es uns wirklich sehr bequem. Und jetzt sehen Sie auch, daß die reservierten Plätze im Ergebnis von den Minuszeichen belegt worden sind. Was bleibt, sind lediglich die Abstände – der guten Lesbarkeit zuliebe.

Das Semikolon ist nicht das einzige Symbol zur Trennung von Variablen hinter einem PRINT-Befehl. Die von Commodore verwendeten Versionen von Basic erlauben auch das Komma – allerdings hat es eine etwas andere Wirkung als das Semikolon.

→ Ersetzen Sie in der Zeile oben, welche den »dreifachen« PRINT-Befehl enthält, die zwei Semikolon durch Komma (direkt durch Überschreiben). <RETURN> nicht vergessen!

Wir sehen jetzt wieder die drei Zahlen, aber viel weiter auseinandergerückt.

Dabei wird der Bildschirm sozusagen in 4 Teile geteilt. Die ausgedruckten Werte beginnen am linken Rand, dann ab dem 10., 20., und 30. Platz.

FAZIT

Hinter einem einzigen PRINT-Befehl können mehrere Werte auf einmal ausgedruckt werden. Sie müssen entweder durch ein Semikolon oder ein Komma getrennt sein.
Das Semikolon bewirkt bei Zahlen einen »normalen« Abstand von zwei Leerstellen.
Das Komma verschiebt die Werte auf das nächste Bildschirm-Viertel.

3.2 Buchstabe und Zeichen ausdrucken

Was macht der PRINT-Befehl mit allen anderen Zeichen und Symbolen? Versuchen Sie es ruhig einmal. Fangen Sie mit den Tasten links oben an und arbeiten Sie sich durch.

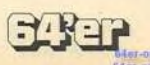








Comic: Ingo Stein



Schon bei dem PRINT-Befehl der allerersten Taste, nämlich dem Symbol »Pfeil nach links« reagiert der Computer unerwartet.

⟶ PRINT ← (<RETURN> drücken)

ergibt eine englische Meldung auf dem Bildschirm:

SYNTAX ERROR
READY

Mit der Meldung »SYNTAX ERROR« sagt uns der Computer, daß wir einen Rechtschreib- oder Satzbaufehler begangen haben. Das Zeichen »←« darf zum Beispiel nicht so einfach mit einem PRINT-Befehl verwendet werden.

Alle mathematischen Zeichen, Interpunktionen, die geSHIFTeten Symbole der Zifferntasten und andere reagieren in dieser Form.

Andere wieder, wie zum Beispiel mit SHIFT- oder Commodore-Taste (CBM-Taste) umgeschalteten Buchstaben (auf die auf der Vorderseite der Tasten angegebenen Grafiksymbole), ergeben gar keine Reaktion.

Den PRINT-Befehl mit Buchstaben haben wir ja schon mehrfach verwendet. In dem Beispiel ganz oben hat er allerdings nicht den Großbuchstaben A, sondern den Zahlenwert ausgedruckt, der unter dem Kennbuchstaben A im Computer gespeichert worden ist.

Versuchen Sie es mit einem anderen Buchstaben oder Wort, welchem wir noch keinen Wert zugewiesen haben, zum Beispiel:

⟶ PRINT WORT (<RETURN> drücken)

Wir erhalten eine Null.

Das heißt nichts anderes, daß der PRINT-Befehl schlicht und einfach jeden Buchstaben bzw. jede Buchstabenfolge als ein Stichwort – als eine Variable – interpretiert und deren Wert im Speicher sucht, um sie auszudrucken. Haben wir ihr noch keinen Wert zugewiesen, wie in unserem letzten Beispiel dem Wort »WORT«, dann druckt er eine Null aus.

Wenn der für den Bildschirm zuständige Editor (ein fest eingebautes Computerprogramm) bereitwillig alle Buchstaben druckt, die wir direkt mit der Tastatur eingeben, dann muß es der Computer mit dem PRINT-Befehl auch können.

Die Lösung für unser Problem könnte man als zungenbrechendes Kochrezept so hinschreiben:

Um dem Computer mitzuteilen, daß das Wort »WORT« ein Wort sein soll, muß man "WORT" schreiben. Das Geheimnis liegt im Gebrauch der Anführungszeichen!

⟶ PRINT "WORT" (<RETURN> drücken)

Für Zeichen, Zahlen, Buchstaben, kurz für alles was zwischen zwei Gänsefüßchen steht, verwendet die Computersprache ein eigenes Wort: *Zeichenkette* oder auf englisch: *String*.

Strings sind demnach auch alle grafischen Symbole der Buchstabentasten, die auf der Vorderseite der Tasten zu sehen sind. Auch sie kann man per PRINT auf den Bildschirm bringen.

Für die linken Symbole müssen wir mit der Taste gleichzeitig die Commodore-Taste (unten ganz links), für die rechten Symbole die SHIFT-Taste drücken.

Die Schwierigkeit für mich liegt nun darin, daß ich Ihnen im Text des Kurses diese speziellen Symbole nicht ohne weiteres angeben kann. Diese Commodore-spezifischen Zeichen und Symbole hat das Fotosatzsystem verständlicherweise nicht zur Verfügung.

Daher werde ich an ihrer Stelle folgende Symbole anwenden:

- für die SHIFT-Taste <SHIFT>
- für die Commodore-Taste <CBM>
- (SHIFT W) bezeichnet das Symbol vorn rechts auf der W-Taste
- <CBMZ> bezeichnet das Symbol vorn links auf der Z-Taste

Ich zeige es am besten an einem Beispiel:

⟶ PRINT "<SHIFT U><SHIFT I>"
(<RETURN> drücken)

dann druckt der Computer aus den beiden Viertelkreisen der U- und I-Taste einen Halbkreis aus.

Noch ein Beispiel:

⟶ PRINT "<CBM Q><CBM W>"
(<RETURN> drücken)

ergibt ein großes H.

Die Schreibweise zwischen Anführungszeichen erlaubt uns, nicht nur Buchstaben und deren umgeschaltete grafische Zeichen, sondern auch Ziffern und alle Symbole auszuPRINTen.

Verzeihen Sie mir bitte dieses saloppe Computerdeutsch. Eigentlich ist es ja nicht sehr schön, was wir da mit der deutschen Sprache machen und ich müßte mich eigentlich schämen, daß ich derartige Wortschöpfungen verwende. Aber abgesehen davon, daß Computerdeutsch gewisse Vorgänge sehr kurz und präzise ausdrückt, verteidige ich mich gern damit, daß andere Gruppen unserer Gesellschaft ja auch ihre eigenen Ausdrücke haben, wie die Segler, Musiker und Jäger. Bei den letzteren meine ich natürlich nicht das Jägerlatein.

Wo waren wir stehen geblieben? Ach ja, beim PRINTen von Ziffern und Symbolen als Strings.

⟶ PRINT "()" (<RETURN> drücken)

Diese Anweisung druckt die beiden Klammern ohne SYNTAX ERROR aus. Alle Pfeile, Sterne, Interpunktionen und arithmetischen Symbole sind willig ausdruckbar, wenn sie in Anführungszeichen eingepackt sind.

A propos Interpunktionen! Wir haben gelernt, daß mit dem Semikolon Stichwörter (Variable) voneinander getrennt werden können, selbst wenn sie hintereinander bei derselben PRINT-Anweisung stehen.

Zwischen Anführungszeichen verlieren sie diese Wirkung; sie sind dann nichts anderes als grafische Zeichen. Geben Sie ein:

⟶ A = 3 (<RETURN>)
⟶ B = 12 (<RETURN>)
⟶ C = 17 (<RETURN>)
⟶ PRINT A;B;C;"A;B;C" (<RETURN>)

Zuerst wird der Wert der drei Variablen A, B und C nebeneinander ausgedruckt, gefolgt von den 5 Symbolen zwischen den Anführungszeichen.

Mit dem Komma und seiner verschiebenden Wirkung ist es ebenso.

Und Ziffern? Ziffern gehen auch, aber sie gingen ja vorher schon, ohne Anführungszeichen.

Da ist aber ein ganz kleiner Unterschied. Passen Sie auf:

⟶ PRINT 123 (<RETURN>)
⟶ PRINT "123" (<RETURN>)

- Der erste PRINT-Befehl versteht die Ziffern 123 als Zahl und reserviert, wie früher schon besprochen, eine Stelle vorher für das Vorzeichen.
- Der PRINT-Befehl mit Anführungszeichen versteht die Ziffern 123 als String, der natürlich kein Vorzeichen braucht und daher auch keine Platzreservierung bekommt.

Dieser feine Unterschied ist recht wichtig, wenn Zahlentabellen oder Zahlengrafiken erzeugt werden sollen. Wir stoßen später sicher noch darauf.

Bevor ich ein weiteres Fazit ziehe, möchte ich Ihnen eine vorläufig letzte Anwendung des PRINT-Befehls zeigen.

Wir haben gesehen (und angewendet), daß eine auf den Bildschirm geschriebene Zeile erst dann zu einem Befehl für den Computer wird, wenn sie mit dem Drücken der RETURN-Taste »abgeschlossen« wird. Wir haben daher immer nur einen einzigen Befehl in eine Zeile geschrieben.

Die Frage stellt sich natürlich:

Können wir mehrere Befehle in
eine einzige Zeile schreiben??

Wie immer – probieren geht über studieren! Geben Sie ein:

→ PRINT "1.BEFEHL" PRINT "2.BEFEHL"
→ drücken Sie die RETURN-Taste

Nun, SYNTAX ERROR sagt uns, daß diese Schreibweise nicht erlaubt ist.

Wie bei mehreren Variablen brauchen wir bei mehreren Befehlen auch ein Symbol zur Trennung. Bei Befehlen ist dies der Doppelpunkt. Fügen Sie bitte zwischen die beiden PRINT-Befehle oben einen Doppelpunkt ein und drücken Sie nochmal auf Return.

Es muß also dastehen:

→ PRINT "1.BEFEHL" : PRINT "2.BEFEHL"

Jetzt stehen die beiden »Befehle« ausgedruckt da – untereinander!

Durch den Doppelpunkt werden die zwei PRINT-Befehle so behandelt, als stünden sie in zwei getrennten Zeilen.

Und damit wird auch unsere anfängliche Rechnerei wesentlich einfacher – ich hoffe, Sie erinnern sich noch. Sonst schauen Sie bitte nochmal am Anfang nach.

Für die Berechnung der Summe 3 + 2 brauchen wir nur eine Zeile:

→ SUMME = 3 + 2 : PRINT SUMME
(<RETURN> drücken)

Damit soll's erstmal genug sein. Ich glaube, Sie haben jetzt eine ganze Reihe von Regeln gelernt, die für die ersten Schritte der Programmierung eines Commodore-Computers sehr wichtig sind.

FAZIT

1. Buchstaben werden vom PRINT-Befehl als Variable eingestuft, und entsprechend wird ihr Wert ausgedruckt.
2. Ziffern und Zahlen werden vom PRINT-Befehl als arithmetische Ausdrücke gewertet und erhalten dementsprechend ein Vorzeichen. Das positive Vorzeichen wird nicht ausgedruckt, eine Leerstelle ist dafür vorhanden.
3. Alle Symbole der Tastatur, welche zwischen Anführungszeichen stehen, werden vom PRINT-Befehl als graphische Zeichen erkannt und formgetreu ausgedruckt. Sie werden Zeichenketten beziehungsweise Strings genannt.
4. Um mehrere Befehle in eine einzige Zeile schreiben zu können, müssen sie durch einen Doppelpunkt voneinander getrennt sein.
5. Schreibfehler oder inkorrekte Anweisungen quittiert der Computer mit der Fehlermeldung SYNTAX ERROR.

Lektion 4: Was ist ein Programm?

Bislang haben wir den Computer so benutzt, als wäre er ein Taschenrechner.

Wir haben alle Angaben, Befehle und so weiter direkt eingetippt; und der Computer hat sie nach dem Drücken der RETURN-Taste direkt ausgeführt.

Ich habe mit Absicht das Wort direkt gleich zweimal verwendet. Diese Art des Betriebs wird nämlich Direkt-Modus genannt.

Im Gegensatz dazu steht der sogenannte Programm-Modus.

Jetzt also taucht das Schlagwort Programm zum ersten Mal auf. Haben Sie schon sehnlich darauf gewartet? Es wäre kein Wunder, wollen wir doch Programmieren lernen.

In meinem alten Konversationslexikon steht unter Programm: Festfolge. Sie wissen, was damit gemeint ist, nämlich eine detaillierte Angabe der einzelnen Darbietungen und Aktivitäten einer Veranstaltung in ihrer genauen zeitlichen Reihenfolge.

Diese Definition läßt sich gut auf einen Computer übertragen.

Wenn ein Computer viele Befehle in einer bestimmten Reihenfolge nacheinander ausführen soll, müssen sie ihm als »Festfolge« vorgegeben sein. Diese Festfolge muß er auswendig kennen – was für ihn kein Problem bedeutet, besitzt er doch ein gutes Gedächtnis.

Dieses »Gedächtnis« haben wir ja schon kennengelernt. Die im Direkt-Modus eingegebenen Werte von Stichwörtern (Variablen) hat er in seinem Speicher festgehalten und wir konnten sie jederzeit abfragen.

Ein Computer hat aber auch einen Programm-Speicher, in welchem er ein aus mehreren Befehlen bestehendes Programm festhält.

Alles, was wir tun müssen, ist, ihm die Befehle und ihre Reihenfolge einzugeben. Wie tun wir das?

Die Angabe der Reihenfolge ist sehr simpel. Wir geben einfach jeder Zeile, in der ein oder mehrere Befehle stehen, eine Nummer.

Die Eingabe der Befehle geht genauso wie vorher – mit der RETURN-Taste. Nur – vorher hat der Computer den Befehl sofort ausgeführt. Wenn aber eine Nummer davorsteht, dann führt er den Befehl nicht aus, sondern steckt ihn in seinen Programmspeicher.

Das machen wir jetzt gleich einmal. Geben Sie bitte unsere allerersten Rechenanweisungen mit Zeilennummern ein:

→ 1 SUMME = 3 + 2 (<RETURN>)
→ 2 PRINT SUMME (<RETURN>)

Siehe da, er hat nichts ausgeführt und ich behaupte, dieses kleine Programm steht jetzt im Programmspeicher.

Ich kann das auch beweisen. Es gibt einen Befehl in Basic, der alles, was im Programmspeicher steht, in der Reihenfolge der Zeilennummern auf dem Bildschirm ausdruckt. Er heißt

LIST

was auf deutsch soviel wie »auflisten« bedeutet.

→ Löschen Sie den Bildschirm mit der CLR-Taste
→ Tippen Sie direkt das Wort LIST ein und drücken Sie die RETURN-Taste.

Die beiden Anweisungen samt Nummer erscheinen auf dem Bildschirm. Was zu beweisen war.

Basic-Befehl Nr. 3 LIST

- druckt den Inhalt des Programmspeichers auf dem Bildschirm aus.
- Durch Angabe der Nummer oder eines Nummernbereiches können einzelne Befehle oder bestimmte Programmteile separat ausgedruckt werden.
- Durch den Befehl LIST wird das Programm nicht gelöscht.

Den ersten Satz dieser Befehlsbeschreibung haben wir ja schon praktiziert. Den zweiten Satz möchte ich vorläufig erst mal so stehen lassen. Zu seiner Erprobung brauchen wir längere Programme und die haben wir vorerst noch nicht.

So, unser Programm, bestehend aus zwei Befehlen, hat der Computer gespeichert. Jetzt soll er es ausführen!

Unsere alte Methode, den Computer hinter dem Ofen hervorzuscheuchen, war der Tritt mit der RETURN-Taste. Diese bringt uns hier aber überhaupt nichts. Was wir brauchen ist ein Befehl, der den Computer veranlaßt, das Programm auszuführen. Ein derartiger Befehl wird uns von Basic geboten. Er heißt

RUN

was mit »loslaufen« übersetzt werden könnte.

Basic-Befehl Nr. 4 RUN
- startet ein im Programmspeicher des Computers befindliches Programm.
- Mit RUN wird ein Programm nicht gelöscht. Der RUN-Befehl kann beliebig oft wiederholt werden.

→ Tippen Sie direkt das Wort RUN ein und drücken Sie die RETURN-Taste.

Und wie der Blitz, so schnell können Sie gar nicht schauen, steht das Resultat auf dem Bildschirm.

Na ja, sagen vielleicht einige Skeptiker unter Ihnen, im Direkt-Modus vorhin ging das auch nicht viel langsamer, besonders die Einzeiler-Version mit dem Doppelpunkt.

Richtig, aber wir haben ja nur zwei Befehle verwendet. Nehmen Sie mal 20 Befehle, da sieht die Sache schon anders aus. Abgesehen von der Geschwindigkeit gehen die überhaupt nicht in eine Zeile – und was dann?

FAZIT
1. Der Computer hat einen Speicher mit mehreren getrennten Bereichen. Alle Variablen und ihre jeweiligen Werte stehen im Variablenspeicher. Programmzeilen stehen im Programmspeicher.
2. Immer wenn die RETURN-Taste gedrückt wird, überprüft der Computer die Zeile, in der sich gerade der Cursor befindet. Beginnt die Zeile mit einer Nummer, wird sie als Programmzeile erkannt und im Programmspeicher abgelegt.
   Hat sie keine Nummer am Anfang, dann führt sie der Computer im Direkt-Modus sofort aus.
3. Mit dem Basic-Befehl LIST werden alle im Programmspeicher stehenden Zeilen in Reihenfolge der aufsteigenden Zeilennummern auf dem Bildschirm ausgedruckt.
4. Ein Programm wird mit dem Basic-Befehl RUN gestartet.

Nun sollten wir uns schleunigst an ein längeres Programm wagen. Natürlich verwenden wir nur Dinge, die wir bisher schon verwendet, beziehungsweise besprochen haben.

Lektion 5: Ein erstes Programmbeispiel

Zuerst brauchen wir einen Plan, was programmiert oder besser gesagt, was vom Computer gemacht werden soll.

Ich schlage dazu eine Aufgabe mit folgenden Teilen vor:
1. Unter dem Stichwort A sollen zwei Zahlen multipliziert werden
2. Unter den Stichwort B sollen zwei Zahlen dividiert werden
3. Das Stichwort C sei die Differenz der Werte von A und von B
4. Der Wert von A soll nicht als Zahl, sondern als Gleichung ausgedruckt werden.
5. Auch der Wert von B soll als Gleichung ausgedruckt werden.
6. Dann folgt eine Textangabe, wie der Wert von C berechnet wird. Dabei sollen im Text die jeweiligen Werte von A und B automatisch eingesetzt werden.
7. Als letztes wird auch der Wert von C als Gleichung ausgedruckt.

Das klingt schlimmer, als es ist. Wir müssen es einfach Schritt für Schritt angehen.

Für den Punkt 1 nehmen wir die Zahlen 24 und 3

→ `1 A = 24 * 3` (<RETURN>)

In dieser Zeile mit der Nummer 1 weisen wir dem Stichwort A das Resultat der Multiplikation von 24 mal 3 zu. Wir hätten das auch mit dem Basic-Befehl LET machen können, aber wie gesagt, man kann ihn weglassen.

Punkt 2 verlangt etwas Ähnliches, wir nehmen die Zahlen 16 und 8.

→ `2 B = 16/8` (<RETURN>)

Auch Punkt 3 macht keine Ausnahme, nur verwenden wir jetzt statt absoluter Zahlen die Stichwörter A und B. Der Computer muß selbst im Lauf des Programms die Werte für A und B im Speicher heraussuchen und einsetzen.

→ `3 C = A - B` (<RETURN>)

Bis hierhin war alles so wie gehabt. Im Punkt 4 soll der Wert von A, also das Resultat aus Zeile 1 als »Gleichung« ausgedruckt werden.

Was ich damit meine, ist schnell gesagt. Der Befehl PRINT A würde nur den Zahlenwert von A auf den Bildschirm bringen. Ich möchte aber, daß das Programm diese 5 Zeichen hinschreibt:

A = 72

Wie erreichen wir das ?

Nun, die Zahl 72 ist der Wert von A, mit PRINT A zu erzielen. Davor steht ein Text! Damit der PRINT-Befehl »A =« hinschreibt, müssen wir das als String schreiben, das heißt, wir müssen den Anführungszeichen-Modus anwenden. Das sieht dann so aus:

→ `4 PRINT "A =" A` (<RETURN>)

Was zwischen den Anführungszeichen steht, druckt der Computer so aus, wie es ist. Das zweite A steht nackt da, also ist es ein Stichwort, dessen Wert der Computer direkt hinter den Text »A =« setzt – mit Vorzeichenstelle natürlich.

Wenn Sie es nicht glauben oder es sich nicht vorstellen können, lassen Sie doch diesen ersten Teil des Programms gleich einmal laufen. Zur besseren Übersicht löschen Sie zuerst den Bildschirm mit der CLR-Taste.

Keine Angst, es wird nur der Bildschirm gelöscht, aber nicht das Programm im Programmspeicher. Um das nachzuprüfen, LISTen Sie es aus. Wenn das Programm schön dasteht, lassen Sie es mit RUN und RETURN-Taste laufen.

Auf Ihrem Bildschirm steht jetzt unter der Programm-Liste:

A = 72

Das ist also das Resultat der Zeile 4. Genauso machen wir es mit dem Punkt 5.

→ `5 PRINT "B = " B` (<RETURN>)

Punkt 6 ist reiner Text, den ich in zwei PRINT-Befehle aufteile, damit der Text in zwei getrennten Zeilen steht.

→ `6 PRINT "C IST A MINUS B,"` (<RETURN>)

→ `7 PRINT "DAS HEISST," A "WIRD UM" B "VERRINGERT."` (<RETURN>)

Zeile 6 besteht aus reinem Text – alles steht zwischen Anführungszeichen. Zeile 7 ist wieder etwas komplizierter, aber sie enthält im Grunde genommen nichts anderes als die Zeilen 4 und 5 vorher. Man muß nur die Anführungszeichen richtig abzählen. Zwischen den ersten beiden wird Text ausgedruckt, inklusive dem Komma. Dann folgt der Wert der Variablen A – ohne Anführungszeichen natürlich. Zwischen den nächsten beiden Textteilen steht wieder einsam das Stichwort B, dessen Wert der Computer suchen und hier einsetzen muß.

Auch jetzt ist es Ihnen schon erlaubt, einen vorläufigen Probelauf zu machen.

Schließlich führen wir noch Punkt 7 aus, wofür wir wieder die Methode der Zeilen 4 und 5 anwenden:

→ `8 PRINT "C =" C` (<RETURN>)

Es empfiehlt sich immer, ein Programm nochmal als Ganzes aufzuLISTen. Also: Bildschirm löschen und LIST eingeben.

Wir sehen dann auf dem Bildschirm:

1 A = 24 * 3

```
2 B = 16/8
3 C = A - B
4 PRINT "A =" A
5 PRINT "B =" B
6 PRINT "C IST A MINUS B,"
7 PRINT "DAS HEISST,"A "WIRD UM" B
"VERRINGERT"
8 PRINT "C =" C
```

Alle diejenigen, welche den PRINT-Befehl mit dem Fragezeichen abgekürzt haben, werden jetzt staunen. Trotz der Abkürzerei steht im Programm-Ausdruck - auch Listing genannt - kein Fragezeichen, sondern das volle Wort »PRINT«. Das ist wieder die Tat des guten alten Editors, der flugs alles in seine rechte Bahn umlenkt und umkodiert.

Wenn Sie beim Tippen keine Fehler gemacht haben, dann dürfen Sie das Programm laufen lassen.

Wenn Sie einen Fehler entdeckt haben, dann nutzen Sie einfach den Editor aus und korrigieren Sie den Fehler direkt in der Zeile, wo er auftritt. Nach der Reparatur nicht die Zeile verlassen, sondern zuerst RETURN drücken, damit die neue, korrigierte Zeile an die Stelle der alten falschen Zeile, die ja dieselbe Nummer hat, gespeichert wird.

So, jetzt hindert uns aber niemand mehr am RUN (und natürlich RETURN-Taste).

Auf Ihrem Bildschirm muß jetzt folgendes Resultat stehen:

```
A= 72
B= 2
C IST A MINUS B,
DAS HEISST, 72 WIRD UM 2 VERRINGERT
C= 70

READY.
```

Ja, was wollen Sie machen, wenn das Programm noch immer nicht das tut, was es soll?

Als allererstes nicht verzweifeln! Tippfehler treten immer wieder auf, die der Computer in seiner Perfektion eben sofort merkt. Da er aber keine eigene Intelligenz hat, tut er wirklich nur genau das, was wir ihm anschaffen - mit allen Fehlern.

Eine Fehlersuche kann oft sehr mühsam sein, besonders bei langen und komplexen Programmen.

Nun, in unserem Fall dürfte es nicht allzu schwer sein, Zeile für Zeile alle Buchstaben und Symbole durchzugehen, bis Sie den Fehler gefunden haben.

Später, in Lektion 34, werden wir noch Methoden der Fehlersuche kennenlernen; jetzt möchte ich Sie noch nicht damit belasten.

Ich möchte vielmehr, daß Sie noch ein bißchen mit dem Programm spielen. Verändern Sie die Zahlenwerte in den Zeilen 1 und 2 durch direktes Überschreiben - aber immer mit <RETURN> abschließen, sonst bleibt der alte Wert erhalten.

Sie brauchen dann nicht erneut RUN eintippen, es steht ja noch da. Fahren Sie mit dem Cursor drauf und drücken Sie die RETURN-Taste. Sofort wird das Programm wieder ausgeführt und wie mit Geisterhand erscheinen neue Zahlen an den Stellen der Variablen.

Aber Achtung!! Die Werte auf dem Bildschirm werden nur überschrieben. Wenn der neue Wert kürzer ist als der alte, dann bleibt der »längere« Rest des alten Wertes stehen; das kann zur, Verwirrung führen.

Das passiert dadurch, daß dem Computer ja keine Anweisung gegeben worden ist, vor der Berechnung den Bildschirm zu löschen. Diese Anweisung kommt erst später in unser Lehrprogramm.

Beachtenswert ist, wie viele Zeichen in einer Programmzeile stehen dürfen. 40 werden Sie vielleicht sagen - aber nein: Eine mit Nummer als Programmzeile gekennzeichnete »logische« Zeile besteht aus 2 »echten« Bildschirmzeilen und enthält somit maximal 80 Zeichen.

Ein anderes »Experiment« besteht darin, die Zeilennummer einer Befehlszeile zu ändern.

→ Fahren Sie mit dem Cursor auf die 8 dieser Zeile Überschreiben Sie die 8 mit der Ziffer 9 und drücken Sie <RETURN>

Wir haben jetzt eine neue Programmzeile geschaffen, mit identischem Inhalt wie die alte Zeile 8.

→ Geben Sie LIST und RETURN ein.

Siehe da, wir haben eine neue Zeile 9, aber die alte Zeile 8 ist auch noch da, natürlich, denn wir haben sie ja durch das Überschreiben nicht gelöscht.

Das Löschen einer Zeile geht ganz einfach:

→ Tippen Sie die Nummer der Zeile, in unserem Falle eine 8 in eine freie Zeile des Bildschirms und drücken Sie <RETURN>

Wir haben also eine Zeile 8 ohne Text beziehungsweise ohne Befehle eingegeben. Da dem Rechner sein Speicherplatz sehr kostbar ist, ignoriert er eine solche nichtssagende Zeile - sie ist weg. Überprüfen Sie es mit LIST.

Jetzt haben wir ein Programm, in dem in der Reihenfolge der Zeilen eine Nummer fehlt. Was macht das? Mit RUN können Sie sich überzeugen, daß das dem Computer völlig egal ist. Er braucht nämlich nur Zeilennummern in aufsteigender Folge, Lücken überspringt er einfach.

Das bedeutet aber, daß wir unser Programm oben auch mit den Zeilennummern 10..20..30.. und so weiter hätten schreiben können. In der Tat möchte ich Ihnen diese Vorgehensweise empfehlen, da sie ein späteres Einfügen von weiteren Programmzeilen erlaubt. Wir werden das noch üben.

FAZIT

1. Innerhalb eines PRINT-Befehls können Variable und Zeichen beliebig miteinander gemischt werden. Wichtig ist nur, daß Zeichen immer innerhalb von Anführungszeichen stehen müssen.
2. Ein bereits im Progammspeicher stehendes Programm kann auf dem Bildschirm jederzeit abgeändert werden. Es gelten bei dem »Überschreiben« alle Regeln und Möglichkeiten des Editors. Eine Änderung muß mit Return abgeschlossen werden.
3. In dem Programmspeicher wird die jeweils letzte mit <RETURN> eingegebene Version einer Zeile gespeichert.
4. Zeilennummern lassen sich auf dieselbe Art und Weise verändern. Eine Zeile wird durch Eingabe lediglich der Zeilennummer, also ohne Text, gelöscht.
5. Zeilennummern brauchen nicht direkt aufeinander zu folgen. Lücken in der Zahlenfolge werden übersprungen.
6. Es empfiehlt sich, ein Programm mit Zeilennummern zu versehen, die zum Beispiel jeweils um 10 voneinander verschieden sind. Das erleichtert das spätere Einfügen von zusätzlichen Programmzeilen.
7. Eine Programmzeile kann maximal 80 Zeichen enthalten. Bei späterem Vergrößern von bestehenden Zeilen schiebt der Editor alle Zeilen automatisch nach unten, um Platz zu schaffen.

Wir haben ein erstes kleines Programm geschrieben. Aber sein Wortschatz ist noch sehr begrenzt - LET und PRINT. Die beiden anderen Befehle, die wir kennen, nämlich LIST und RUN werden nur im Direkt-Modus verwendet. Bevor wir weitermachen, empfehle ich Ihnen, das alte Programm aus dem Speicher des Computers zu entfernen und ihn frei zu machen für ein neues Programm.

Verwechseln Sie das bitte nicht mit dem Löschen des Bildschirms. Zur Klärung:

- Die CLR/HOME-Taste geSHIFTet löscht den Bildschirm; ein Programm im Speicher bleibt davon unberührt.
- Durch Aus- und Wiedereinschalten des Computers löschen Sie sowohl den Bildschirm als auch den Speicher des Computers, in welchem das Programm gespeichert ist. Der Computer meldet sich dann mit seinen Einschalt-Überschriften.
- Es gibt auch einen Basic-Befehl, der ein Programm aus dem Speicher hinauswirft, ohne den Bildschirm zu löschen. Er heißt

NEW

Probieren Sie es bitte aus.

⟶ Geben Sie eine beliebige Programmzeile (mit Nummer) ein, zum Beispiel:

```
20 PRINT"ABCDE"
```

⟶ löschen Sie mit der CLR-Taste den Bildschirm geben Sie direkt LIST ein

Jetzt erscheint die obige Zeile 20 wieder.

⟶ geben Sie direkt NEW ein und nach der RETURN-Taste wieder den Befehl LIST

Die Programmzeile 20 ist verschwunden

Basic-Befehl Nr. 5 NEW
- macht den Programmspeicher frei für die Eingabe eines neuen Programms
- ein »altes« im Speicher sitzendes Programm wird dadurch zwar nicht gelöscht, aber es kann nach dem NEW-Befehl nur noch unter ganz bestimmten Umständen und nur mit besonderen Tricks wieder verfügbar gemacht werden

Seien Sie also vorsichtig mit diesem Befehl, immer erst überlegen, bevor Sie NEW mit der RETURN-Taste abschließen, hat er doch eine so gut wie endgültige Wirkung. Salopp ausgedrückt »löscht« er den Programmspeicher.

Auch dieser Befehl wird fast ausschließlich nur im Direkt-Modus eingesetzt.

Lektion 6: Mehr über Stichwörter/Variable

Ich habe Ihnen erklärt, daß wir mit Variable die Stichwörter bezeichnen, unter denen der Computer sich Zahlenwerte merkt und unter denen sie auch wieder aus dem Speicher hervorgeholt werden können.

Des öfteren wird das alles mit Schachteln verglichen, auf die der Name einer Variablen geschrieben wird. Ein Wert, welcher der Variablen zugeordnet wird, kommt in die Schachtel mit dem richtigen Namen hinein. Eine Kopie des Wertes kann jederzeit aus der Schachtel herausgeholt werden, wenn man ihren Namen kennt. Der Wert bleibt solange in der Schachtel, bis er durch einen neuen ersetzt wird.

Es gibt mehrere Arten von Variablen, von denen wir zwei besprechen wollen.

6.1 Numerische Variable

Diesen Typ haben wir bislang verwendet. Er enthält nur Zahlen: ganze Zahlen (325), Dezimalbrüche (0.325) und negative Zahlen (-325, -0.325).

Bei der Wahl des Namens der Variablen, der vorn auf die Schachtel geschrieben wird, sind wir ziemlich frei. Er muß nur immer mit einem Buchstaben anfangen. Die nachfolgenden Zeichen können Buchstaben, Ziffern oder grafische Zeichen sein.

Wir haben bisher X, SUMME, PRODUKT etc. verwendet. Wir dürfen aber auch B23X oder F5 nehmen.

Der Länge des Variablen-Namens ist theoretisch nur durch die Anzahl der verfügbaren Zeichen in der Programmzeile eine Grenze gesetzt. Praktisch ist das aber ohne Bedeutung, da der Computer nur die ersten beiden Zeichen als Namen verwendet. So vielsagend zum Beispiel in einem Programm die Variablen-Namen »PRODUKT« und »PROFIT« sein könnten, der Computer erkennt nur »PR« und nimmt an, es handle sich um dieselbe Variable. Also Vorsicht !!!

6.2 String-Variable

So nennen wir den zweiten Typ. Hier geht es nun um Schachteln, in die wir Buchstaben, Zeichen, Wörter, ja sogar ganze Sätze bis zu einer maximalen Länge von 255 Zeichen hineingeben dürfen.

Diese Aneinanderreihung von Zeichen bezeichnen wir, wie schon erwähnt, mit »Zeichenketten« oder mit dem englischen Wort »String«, das sich wegen seiner Kürze allgemein durchgesetzt hat. Daher auch der Name »String-Variable«.

Der Name einer String-Variablen ist genauso aufgebaut wie der einer numerischen Variablen, nur muß am Ende immer das Dollar-Zeichen »$« – die geSHIFTete 4-Taste – stehen.

Der Vergleich der beiden Variablentypen sieht so aus:

```
10 A = 25
20 A$ = "ZEICHENKETTE"
30 PRINT A
40 PRINT A$
```

Sie sehen, selbst bei gleichem Namen, nämlich A, unterscheidet der Computer exakt zwischen der numerischen Variable A und der Stringvariable A$.

Strings und Stringvariable werden uns fortan laufend begegnen. Deswegen schlage ich zur Übung noch ein paar Beispiele vor.

Löschen Sie bitte das obige Programm mit NEW und geben Sie die folgenden Programmzeilen ein, wobei Sie genau auf Semikolons und Doppelpunkte achten müssen.

```
NEW
10 T$="HOLZ"
20 U$="FEUER"
30 W$="7"
40 X$="5"
50 PRINT U$:PRINT T$
60 PRINT U$;:PRINT T$
70 PRINT T$;:PRINT U$
80 PRINT W$;:PRINT X$
90 PRINT W$ X$
```

Diese Programmzeilen ergeben auf dem Bildschirm folgendes Bild:

```
FEUER
HOLZ
FEUERHOLZ
HOLZFEUER
75
75
```

In den Zeilen 10 bis 40 werden 4 String-Variable definiert und ihnen werden zwei Wörter und zwei Ziffern zugewiesen. Die Ziffern sind nicht Zahlenwerte, sondern ebenfalls Strings, da sie zwischen Anführungszeichen stehen.

Zeile 50 enthält zwei – durch den Doppelpunkt getrennte – PRINT-Befehle, die untereinander die Strings der beiden Variablen U$ und T$ ausdrucken.

Zeile 60 ist fast identisch mit Zeile 50, und doch ist ihr Resultat grundlegend verschieden, da beide Strings nebeneinander in einem Wort geschrieben werden. Der winzige Unterschied ist das Semikolon am Ende des ersten PRINT-Befehls. Wir haben das Semikolon schon kennengelernt, als Trennzeichen bei der Aneinanderreihung von mehreren numerischen Variablen innerhalb eines einzigen PRINT-Befehls.

Hier bei den Strings hat es eine entgegengesetzte Wirkung. Es klebt nämlich die Strings zweier Variablen, die in getrennten PRINT-Befehlen stehen, aneinander.

Ich weiß, das sieht verwirrend aus. Aber glauben Sie mir, diese »Rechtschreibregeln« werden Ihnen sehr rasch in Fleisch und Blut übergehen.

Zeile 70 ist ein weiteres Beispiel für den Alleskleber Semikolon, nur diesmal in umgekehrter Reihenfolge der Strings.

Zeile 80 zeigt Ihnen, daß die Ziffern 7 und 5 in der Tat nicht als Zahlenwerte, sondern als Strings behandelt werden. Numerische Variable werden , wie Sie sich erinnern, mit Leerzeichen für das Vorzeichen und für den Abstand ausgedruckt, Strings dagegen nicht. Und da ist es egal, ob der String zufällig eine Ziffer ist.

Zeile 90 schließlich zeigt, daß wie bei den numerischen Variablen auch mehrere String-Variable hinter einen einzigen PRINT-Befehl geschrieben werden können. Da das $-Zeichen das Ende der String-Variablen eindeutig festlegt, kann bei Strings das Semikolon weggelassen werden.

Die nächsten drei Zeilen erweitern das Programm:

```
→ 100 Y$=T$+U$
  110 Z$=U$+T$
  120 PRINT Y$:PRINT Z$
```

Die beiden String-Variablen T$ und U$ werden zu einem neuen String Y$ und in umgekehrter Reihenfolge zu Z$ zusammengebaut (Zeile 100 und 110). Zeile 120 druckt uns das Resultat aus.

Zum Schluß will ich noch meine Behauptung von oben, daß nämlich nur die beiden ersten Zeichen einer Variablen erkannt werden, beweisen.

```
→ 130 PRODUKT=25:PRINT PR
  140 PROFIT= 3:PRINT PR
  150 PRINT PRODUKT
```

Zeile 130 weist der numerischen Variablen »PRODUKT« den Zahlenwert 25 zu und druckt ihn aus, wobei die ersten beiden Buchstaben der Variablen genügen, um sie im Speicher zu finden.

Zeile 140 tut dasselbe für die Variable »PROFIT« mit dem Zahlenwert 3. Und siehe da, die Variable PR, die vorher noch 25 war, wird jetzt mit dem Wert 3 ausgedruckt. Daß wir tatsächlich in Zeile 140 der Variablen PRODUKT einen neuen Wert zugewiesen haben, obwohl wir den Variablen-Namen PROFIT gewählt haben, beweist uns Zeile 150, die unter PRODUKT dennoch die 3 ausdruckt.

Übrigens, wenn Sie nur der Ablauf der letzten drei Zeilen des Programms interessiert, können Sie eine Eigenschaft des Befehls RUN ausnützen:

Wenn Sie

```
RUN 130
```

eingeben, läuft das Programm ab Zeile 130 los und läßt alle vorhergehenden Zeilen unbeachtet.

Wenn Sie aber nach dem RUN eine Zeilennummer angeben, die im Programm nicht vorkommt, läuft das Programm nicht los, sondern es meldet sich der Computer mit der Fehlermeldung »UNDEF'D STATEMENT« (undefined statement), was soviel heißt wie »Zeilennummer nicht definiert«.

FAZIT

1. Wir haben bisher zwei Typen von Variablen kennengelernt:
   - den »numerischen« Variablen weisen wir ausschließlich Zahlenwerte zu
   - den Stringvariablen, gekennzeichnet durch das $-Zeichen am Ende des Variablennamens, weisen wir ausschließlich Zeichen, Buchstaben oder ganze Folgen von ihnen zu. Diese Zeichen- und Buchstabenfolgen heißen »Zeichenketten« oder »Strings«.

   Zahlen sind auch zugelassen, nur werden sie wie Zeichen (und nicht als Zahlenwerte) behandelt.
2. Die Zuordnung von Zahlen oder Strings zum falschen Variablentyp führt generell zu einer Fehlermeldung und oft auch zum Abbruch des Programms.
3. Das Semikolon am Ende eines PRINT-Befehls bewirkt, daß der nächste PRINT-Befehl, egal wo er steht, seinen Text direkt anschließend zum vorhergehenden druckt.
4. Um ein Programm ab einer bestimmten Zeilennummer zu starten, wird diese Zeilennummer dem RUN-Befehl angehängt (RUN 600).

LEKTION 7: Sprünge im Programm

Bislang bestand ein Programm aus einzelnen Zeilen, die der Reihe nach abgelaufen sind.

Ein Programm muß nicht so einfach sein. Es vermag auch Sprünge auf beliebige Zeilen durchzuführen.

Dazu enthält die Sprache BASIC einen Befehl, der den Computer hüpfen läßt, und zwar auf eine mit diesem Befehl angegebene Programmzeile. Der Befehl lautet:

GOTO (Zeilennummer)

Er darf auch in der Form GO TO geschrieben werden. Auf deutsch bedeutet er GEHE NACH.

Durch ihn veranlaßt, springt der Computer auf die angegebene Programmzeile und fährt dort mit dem Programm fort.

Der Befehl GOTO darf auch im Direkt-Modus verwendet werden.

Ich nehme an, Sie haben noch das letzte Programm mit den Strings (Zeile 10 bis 150) im Computer. Wenn Sie jetzt direkt eintippen:


```
→ GOTO 130
```

springt der Computer auf die Zeile 130 und führt sie und die nachfolgende Zeile aus. Im Direkt-Modus wirkt er also wie der RUN-Befehl.

Im Programm-Modus sieht der Befehl genauso aus, er ist halt nur mit einer Zeilennummer versehen:

```
→ 65 GOTO 130
```

Durch den Sprungbefehl in der neuen Zeile 65 rückt das Programm auf die Zeile 130 vor und überspringt die Zeilen 70 bis 120.

Der GOTO-Befehl kann als Zieladresse aber auch eine niedrigere Zeilennummer als er selbst haben. Dann springt er im Programm zurück und bildet so eine Schleife. Geben Sie bitte ein:

```
→ 80 GOTO 60
```

Dieser Sprungbefehl wiederholt die beiden Zeilen 60 und 70 endlos lange oder aber bis Sie die STOP-Taste drücken.

Basic-Befehl Nr. 6 GOTO

- wird mit der folgenden Schreibweise verwendet:
  GOTO (Zeilennummer)
- veranlaßt den Computer sowohl im Direkt- als auch im Programm-Modus, auf die hinter dem Befehlswort stehende Zeilennummer zu springen
- erlaubt Sprünge sowohl auf höhere, als auch auf niedrigere Zeilennummern; durch Rücksprünge können Programm-Schleifen erzeugt werden
- fehlt die Zeilennummer des GOTO-Befehls im Programm, bleibt das Programm mit der Fehlermeldung »UNDEF'D STATEMENT ERROR« mit Angabe der Zeilennummer des falschen Befehls stehen.

Die letzte Anwendung des GOTO-Befehls hat eine sogenannte Endlos-Schleife erzeugt, aus der ein Computer von allein nicht mehr herauskommt. Sie ist natürlich ziemlich

sinnlos, denn außer zum Füllen des Bildschirms mit stets denselben Zeichen nützt sie uns gar nichts.

Da aber Programmschleifen sehr nützliche Werkzeuge sein können, soll ihnen eine eigene Lektion gewidmet sein.

LEKTION 8: Schleifen und Prüfungen

Mit Schleifen können wir zum Beispiel zählen. Das will ich Ihnen zeigen. Geben Sie ein:

```
→ NEW
  10 X=X+1
  20 PRINT X
```

Nach RUN druckt uns die Zeile 20 eine 1 aus. Am Anfang des Programms ist der Variablen X noch keine Zahl zugeordnet, also ist sie 0. In Zeile 10 wird 1 dazugezählt, was 1 ergibt.

Das ist beileibe nicht trivial, denn jetzt ergänzen wir das Programm mit:

```
→ 40 GOTO 10
```

Die dadurch gebildete Schleife wiederholt den Vorgang, und in Zeile 10 wird bei jedem Durchgang die Variable X um 1 erhöht.

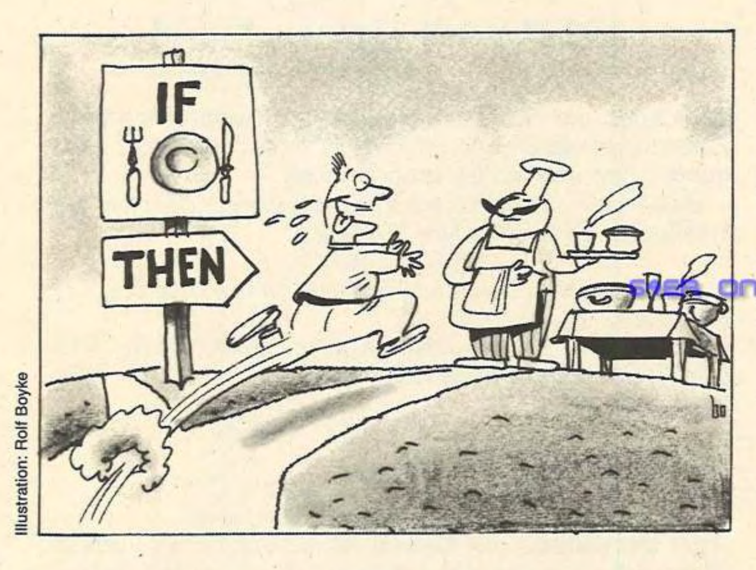


Illustration: Rolf Boyke

Dieses kurze Programm erzeugt eine Zählschleife, die eine endlose Zahlenkolonne über den Bildschirm sausen läßt. Mit der CTRL-Taste können Sie diesen Lauf verlangsamen, mit der STOP-Taste abbrechen.

Diese endlosen Schleifen sind aber wie Autobahnen ohne Ausfahrt. Man kommt leicht drauf, und wer drauf ist, kommt nicht runter.

Wir brauchen also eine Methode, die den Aussprung aus einer Schleife ermöglicht. Dabei wollen wir aber angeben können, wann oder wo wir ausspringen.

In einer Zählschleife wäre das denkbar beim Erreichen einer bestimmten Zahl, zum Beispiel 15.

Prüfung ohne Noten

Basic kennt einen Befehl, der prüft, ob eine bestimmte, von uns festlegbare Bedingung erfüllt ist. Dann nämlich tut er, was man ihm vorgegeben hat.

Der Befehl besteht aus den beiden Wörtern

IF THEN

auf deutsch WENN DANN.

Die volle Schreibweise sieht so aus:

```
IF (Prüfbedingung) THEN (Aktion)
IF     X=3         THEN  PRINT "DREI"
```

X=3 ist die Prüfbedingung, PRINT "DREI" die Aktion.

Der Befehl prüft also, ob die Prüfbedingung erfüllt ist oder nicht. Man sagt auch, er prüft, ob sie »wahr« oder »falsch« ist.

- ist sie erfüllt, dann wird die hinter dem THEN stehende Aktion ausgeführt
- ist sie nicht erfüllt, dann wird ungeachtet dessen, was noch in der Zeile steht, das Programm mit der nächsten Zeile fortgesetzt.

Die Arbeitsweise sehen Sie am besten in unserem Beispiel. Fügen Sie bitte den Zeilen 10 und 20 die zwei folgenden Zeilen 30 und 50 hinzu:

```
→ 10 X=X+1
  20 PRINT X
  30 IF X=15 THEN GOTO 50
  40 GOTO 10
  50 PRINT"ENDE"
```

Also, in Zeile 10 wird X zu 1, Zeile 20 druckt die 1 aus.

Zeile 30 prüft, ob X bereits den Wert 15 erreicht hat. Dies ist nicht der Fall, deswegen geht das Programm in der nächsten Zeile – nämlich 40 – weiter. Zeile 40 springt auf Zeile 10 zurück, X wird um 1 erhöht und hat jetzt den Wert 2. Die Bedingung in Zeile 30 ist aber noch immer nicht erfüllt. Also wiederholt sich der ganze Vorgang, solange, bis die Bedingung des IF-Befehls erfüllt ist. Erst wenn X den Wert 15 erreicht hat, tritt der THEN-Teil des Prüfbefehls in Kraft und führt das aus, was hinter dem THEN steht. In unserem Beispiel springt er mit GOTO 50 aus der Schleife heraus auf die Zeile 50.

In Zeile 50 endet das Programm mit dem Ausdruck »ENDE«.

Man muß bei der Festlegung der Prüfbedingung aufpassen, daß sie überhaupt auch auftritt. Wenn wir zum Beispiel in Zeile 10 den Wert fürr X jeweils um 2 erhöhen, also die Zeile 10 so schreiben:

```
→ 10 X=X+2
```

ist die Schleife wieder endlos, weil 14 oder 16 erreicht wird, aber niemals 15.

Das Problem ist lösbar mit einer Prüfung, ob X größer als 15 geworden ist. Dazu existiert ein Zeichen »>«, das mit der geSHIFTeten Punkt-Taste erzeugt wird.

```
→ 30 IF X > 15 THEN GOTO 50.
```

Jetzt wird die Zeile erst bei der Zahl 16 verlassen.

Als Prüfbedingung stehen uns mehrere mathematische und logische Ausdrücke zur Verfügung:

ist gleich	=	ist ungleich	< >
kleiner	<	Boole'sches UND	AND
größer	>	Boole'sches ODER	OR
kleiner oder gleich	< =	Negation	NOT
gleich oder größer	= >		

Die ersten 6 davon sprechen für sich selbst, die letzten 3 werde ich in Lektion 31 behandeln.

Mit der Ungleich-Bedingung können wir unsere Schleife sogar noch viel eleganter gestalten:

```
→ 30 IF X <> 15 THEN GOTO 10
  40 PRINT "ENDE"
  50
```

Wir sparen Zeile 50 ein, weil die Prüfung immer erfüllt ist, bis X den Wert 15 erreicht hat.

Für den IF-THEN-Befehl sind mehrere Schreibweisen erlaubt:

1. Möglichkeit

Zeile 30 dürfen wir vereinfacht schreiben:

```
30 IF X <> 15 THEN 10
```

oder

```
30 IF X <> 15 GOTO 10
```

Anstelle von THEN GOTO kann auch THEN oder GOTO allein stehen.

2. Möglichkeit

Nach dem THEN kann auch ein anderer Basic-Befehl stehen:

```
110 PRINT X
120 IF X <> 15 THEN X=X+1
130 GOTO 110
140 PRINT "ENDE"
```

In Zeile 120 wird die Zählvariable X direkt weitergezählt, womit derselbe Effekt erzielt wird wie im Programmteil vorher (Zeile 10 bis 50).

Nur mit dem ENDE klappt es noch nicht. Sobald nämlich X=15 ist, gilt die Bedingung in Zeile 120 nicht mehr - und die Zeile 130 kommt wieder an die Reihe, was wir ja nicht wollen.

Um das zu vermeiden, muß die Zeile 120 so lauten:

```
→ 110 PRINT X
  120 IF X <> 15 THEN X=X+1:GOTO 110
  130    (entfällt)
  140 PRINT "ENDE"
```

Sie müssen also immer aufpassen und an die IF-THEN-Regeln denken:

- wenn IF »wahr« ist, wird alles hinter THEN ausgeführt
- wenn IF »falsch« ist, kommt die nächste Zeile zum Zuge

3. Möglichkeit

Mehrere Zählschleifen können unabhängig voneinander gleichzeitig ablaufen. Im folgenden Beispiel beginnt die erste Schleifenvariable X ab dem Wert 0 und wird bei jedem Durchlauf um 1 erhöht. Die zweite Schleifenvariable Y beginnt ab 28 und wird stetig um 2 reduziert:

```
→ 210 X=0:Y=28
  220 X=X+1
  230 Y=Y-2
  240 PRINT X;Y
```

Als Bedingung für das Ende wähle ich den Fall, wo X größer als Y geworden ist:

```
→ 250 IF X < Y THEN 220
  260 PRINT "ENDE"
```

Sobald X=10 und Y=8, ist die Prüfbedingung der Zeile 250 erfüllt und das Programm bleibt stehen.

4. Möglichkeit

Mehrere Zählvariable können auch voneinander abhängig sein. Das folgende Programmbeispiel zeigt das auf einfache Weise. Obwohl diese verschachtelte Schleife keinen praktischen Wert hat, zeigt sie den Zusammenhang doch sehr eindrucksvoll:

```
→ 310 X=2:Y=1
  320 X=X+Y-2
  330 Y=Y-X+3
```

In Zeile 310 stehen die Anfangswerte der Zählvariablen, Zeile 320 und Zeile 330 enthalten die Formeln, nach denen in jedem Umlauf der Schleife die Werte von X und Y »weitergezählt« werden.

Um das Resultat zu sehen, brauchen wir noch zwei Zeilen:

```
→ 340 PRINT X;Y
  350
  360 GOTO 320
```

X und Y nehmen folgende Werte an:

X	2	1	2	4	5	4	2	1
Y	1	3	4	3	1	0	1	3

usw.

Man sieht, daß nach 6 Umläufen der Schleife die ursprünglichen Werte von X und Y wieder auftreten.

Als Prüfbedingung erlaubt uns diese Schleife eine interessante Variante. Für X kommt die Zahl 4 gleich zweimal vor, einmal bei Y=3 und noch einmal bei Y=0. Diese zweite Kombination wollen wir abfragen. Dazu nützen wir eine weitere Eigenschaft des IF-THEN-Befehls aus:

Durch Hintereinanderstellen von mehreren IF-THEN-Befehlen kann eine Mehrfach-Prüfung erzielt werden.

```
→ 350 IF X=4 THEN IF Y=0 THEN PRINT "ENDE"
```

Im dritten Umlauf der Schleife ist die erste IF-Bedingung bereits erfüllt, aber nicht die zweite. Erst nach dem 5. Umlauf bleibt die Schleife stehen.

Basic-Befehl Nr. 7 IF..THEN

- nach dem Befehlswort IF steht eine Prüfbedingung, nach dem Befehlswort THEN eine Aktionsanweisung
- IF prüft, ob die Prüfbedingung erfüllt ist, wenn ja, dann wird die Aktionsanweisung und der Rest der Zeile ausgeführt, wenn nein, läuft das Programm mit der nächsten Zeile weiter
- mehrere IF..THEN-Befehle können hintereinander gestellt eine Mehrfach-Prüfung bilden
- bei IF..THEN GOTO (Zeilennummer) kann entweder das THEN oder das GOTO weggelassen werden

Schleifen mit eigenem Befehl

Wir haben die Schleifen bis jetzt mit Hochzählen einer Schleifenvariablen und Prüfen, ob die Aussprungsbedingung schon erreicht ist, programmiert.

Basic stellt uns dafür einen sehr bequemen Befehl zur Verfügung. Er besteht aus 3 Wörtern:

FOR..TO..NEXT

Auf deutsch läßt sich das übersetzen mit »Für..Bis..Der Nächste«.

Ein Beispiel soll ihn erklären. Links steht die alte Zählschleife, rechts die neue Befehlsfolge.

```
→ 10 X=X+1              10 FOR X=1 TO 10
  20 PRINT X            20 PRINT X
  30 IF X=10 THEN END
  40 GOTO 10            40 NEXT X
```

Man sieht sehr schön den Zusammenhang.

Die alte Methode der Definition einer immer um 1 höherzählenden Variablen mitsamt einer Prüfung ist jetzt in einer Zeile mit den Befehlsteilen FOR TO zusammengefaßt. Die Prüfung ist eingebaut.

Dem alten Rücksprung mit GOTO entspricht jetzt der Befehlsteil NEXT.

Die FOR-NEXT Schleife hat immer die folgende Schreibweise:

```
FOR    Schleifenvariable = Anfangswert
TO     Endwert
STEP   Schrittweite
       .
       .
       (Programm innerhalb der Schleife)
NEXT   Schleifenvariable
```

Die Schleife

```
FOR T=0 TO 14 STEP 2
```

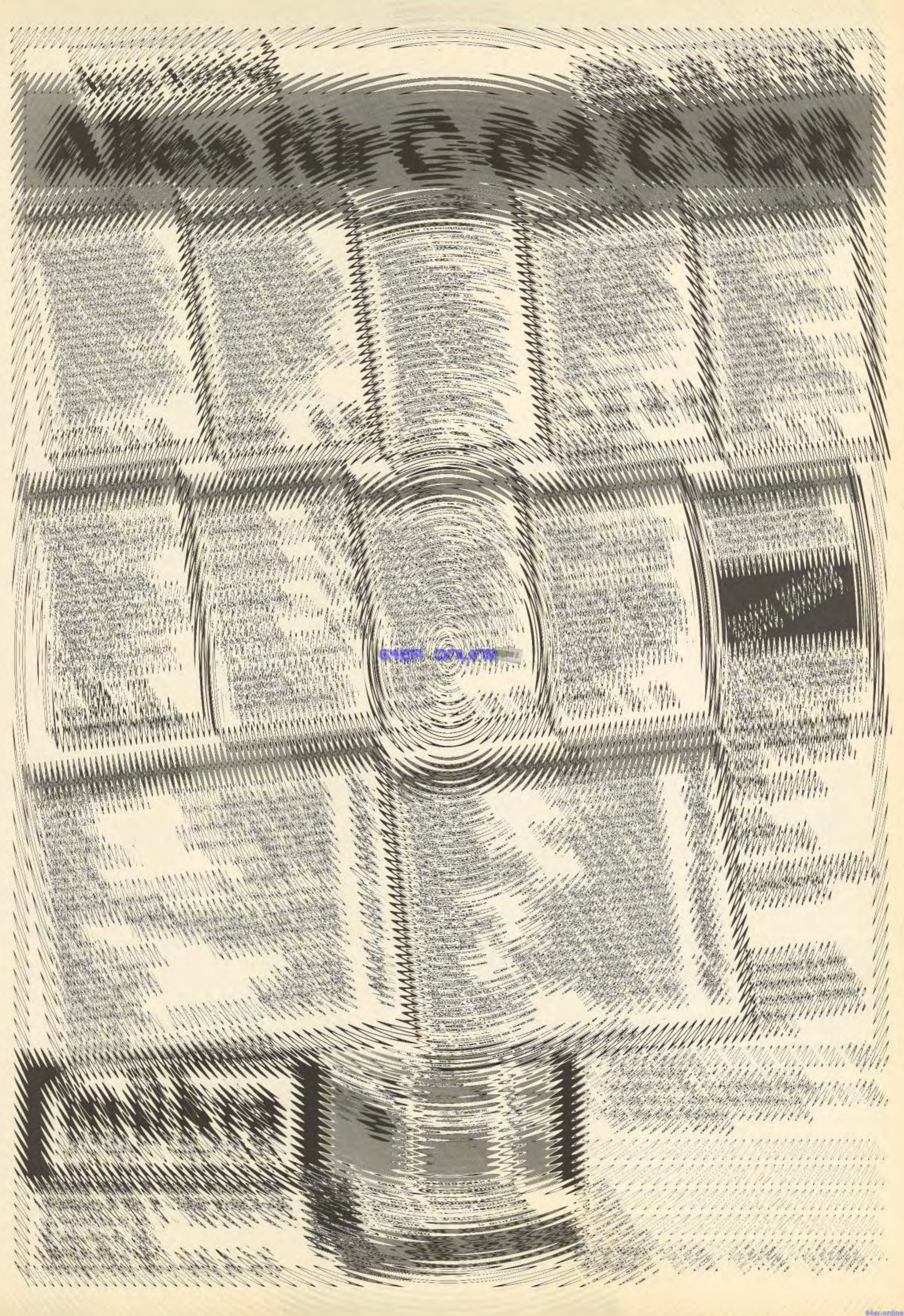
setzt also die Schleifenvariable T auf Null und zählt bis 14 hoch, allerdings immer in Zweierschritten. Wenn STEP weggelassen wird, dann wird automatisch die Schrittweite 1 genommen

Eine rückwärtszählende Schleife sieht so aus:

```
FOR T=14 TO 0 STEP -1
```

In diesem Fall muß die Schrittweite, weil negativ, immer angegeben werden.

Interessant ist auch, daß die Schrittweite ein Dezimalbruch sein kann.



```
→ 10 FOR X=1 TO 15 STEP 3.7
  20 PRINT X
  30 NEXT X
```

ergibt die Zahlen 1, 4.7, 8.4, 12.1 - also jeweils um 3.7 erhöht.

Daraus ist ersichtlich, daß der NEXT-Befehl in der Schleife nicht prüft, ob die Schleifenvariable X den Endwert 15 exakt erreicht hat, sondern ob sie größer ist.

Auch der FOR-TO-NEXT-Befehl erlaubt uns, Schleifen zu »schachteln«. Was das bedeutet, zeigt uns das nächste Beispiel:

```
→ 10 FOR X=1 TO 3
  20        FOR Y=X TO X+2
  30        PRINT Y
  40        NEXT Y
  50 NEXT X
```

Die Zeilen 10 und 50 bilden die äußere Schleife. Die Zeilen 20 bis 40, die ich zur besseren Übersicht weiter eingerückt habe, bilden die innere Schleife.

Wir wollen nachvollziehen, was da abläuft.

Sie sehen, daß die innere Schleife so oft wiederholt wird, wie es die äußere Schleife vorgibt.

Der NEXT-Befehl hat noch zwei Feinheiten. Bei einer einzigen Schleife kann die Angabe der Schleifenvariablen hinter ihm wegfallen. Die folgende kleine Zeitverzögerungsschleife demonstriert das:

```
→ FOR T=1 TO 500: NEXT
```

Bei geschachtelten Schleifen stehen oft mehrere abschließende NEXT-Befehle hintereinander. Statt:

```
  40 NEXT Y
  50 NEXT X
```

kann geschrieben werden:

```
  40 NEXT Y,X
```

Dabei ist auf die Reihenfolge der Schleifenvariablen zu achten. Zuerst kommt immer die innere Variable.

Neben allen diesen geschilderten Feinheiten hat die FOR-TO-NEXT-Schleife noch einen weiteren Vorteil gegenüber der Zählschleife mit IF-THEN: Sie ist wesentlich schneller in der Ausführungszeit.

Basic-Befehl Nr. 8 FOR-TO STEP-NEXT

- hinter FOR steht die Schleifenvariable mit ihrem zugewiesenen Anfangswert. Hinter TO steht der Endwert der Schleifenvariable
- mit STEP kann die Schrittweite eingestellt werden. Sie kann auch negativ oder ein Dezimalbruch sein
- rückwärts zählende Schleifen müssen immer eine negative Schrittweite mit STEP angeben
- mehrere Schleifen können geschachtelt werden. Die innere Schleife muß vor der äußeren abgearbeitet werden
- bei NEXT kann die Schleifenvariable weggelassen werden, wenn dadurch keine Zweideutigkeiten mit anderen Schleifenvariablen entstehen
- wenn mehrere NEXT-Befehle hintereinander auftreten, können ihre Schleifenvariablen durch Kommata getrennt hinter einem einzigen NEXT stehen. Die Reihenfolge »innere Variable vor äußerer Variable« ist einzuhalten

Das folgende Beispiel soll noch einmal die Zählschleife mit IF-THEN der Version mit FOR-TO-NEXT gegenüberstellen und diesen interessanten Unterschied aufzeigen.

Mit beiden Methoden wollen wollen wir 20 Zeilen des Bildschirms mit dem Buchstaben A füllen

```
→ 10 PRINT "(CLR-Taste)"
  20 X=X+1
  30 PRINT "A";
  40 IF X<> 20*40 THEN 20
```

Zeile 10 löscht den Bildschirm. Zeile 20 ist der Zähler. In Zeile 30 dürfen Sie das Semikolon nicht vergessen, damit alle Buchstaben nebeneinander geschrieben werden.

Die Prüfbedingung in Zeile 40 fragt nach, ob 20 Zeilen mal 40 Plätze gefüllt sind, eine Schreibweise, die durchaus erlaubt ist.

Schreiben Sie die andere Version direkt darunter:

```
→ 110 PRINT "(CLR-Taste)"
  120 FOR X=1 TO 20*40
  130 PRINT "A";
  140 NEXT X
```

Diese Zeilen brauche ich Ihnen nicht erklären.

Der frappierende Unterschied:

- die IF-THEN-Schleife benötigt ca. 10 Sekunden
- die FOR-TO-NEXT-Schleife benötigt nur wenig mehr als 2 Sekunden

Auf dieses Phänomen der Laufzeitunterschiede werde ich in Lektion 23 noch eingehen.

LEKTION 9: Eingabe

So, jetzt möchte ich Sie mit der Möglichkeit vertraut machen, während des Ablaufs eines Programms - also nicht schon beim Eintippen desselben - dem Computer Anweisungen in Form von Zahlen und Wörtern zu geben.

9.1. Eingabe mit Input

Bislang haben wir, wie gesagt, alle Zahlen und Wörter, die der Computer auf den Bildschirm bringen sollte, immer vorher, das heißt, beim Schreiben der Programmzeilen eingegeben, als Wert-Zuweisung für eine Variable (das gute alte Stichwort), mit und ohne LET-Befehl.

Dasselbe während des Ablaufs eines Programms ermöglicht der BASIC-Befehl

INPUT (gefolgt von einer Variablen)

Auf deutsch würde dieser Befehl »Eingabe« heißen. Im Direkt-Modus können wir ihn leider nicht einsetzen. Versuchen Sie es ruhig:

→ INPUT A (RETURN nicht vergessen!)

Der Computer bestraft uns mit der Fehlermeldung »ILLEGAL DIRECT ERROR«. Aber im Programm-Modus, das heißt mit einer Zeilennummer versehen, geht es. Auch das wollen wir ausprobieren:

```
→ 10 INPUT A
```

Jetzt nur noch RUN eingeben, und siehe da, der Computer druckt ein Fragezeichen und wartet mit blinkendem Cursor.

Daß er wirklich wartet, merken Sie daran, daß er beharrlich immer wieder die Fehlermeldung »REDO FROM START« - was einfach »NOCH EINMAL« heißt - und danach das auffordernde Fragezeichen ausdruckt, wenn Sie einen Buchstaben oder ein Zeichen eingeben.

Er akzeptiert nur Zahlen, und außerdem noch die Leertaste, den Punkt, das Plus- und das Minuszeichen. Die RETURN-Taste bricht den Befehl ab, und das Programm fährt mit der nächsten Zeile fort.

Warum akzeptiert INPUT A nur Zahlen? Nun, was passiert bei INPUT eigentlich?

Die Zahl, die wir per Tastatur eingeben, wird der Variablen A, die hinter dem Befehlswort INPUT steht, zugeordnet, genauso wie mit dem LET-Befehl.

Mit der folgenden Programmzeile, die Sie zusätzlich zur Zeile 10 eingeben, wird diese Zuordnung bewiesen:

```
→ 10 INPUT A
  20 PRINT A
```

Die beiden Zeilen 10 und 20 zusammen ergeben ein Mini-Programm, welches nach RUN auf eine Eingabe wartet und diese - sofern es eine Zahl ist - auf dem Bildschirm ausdruckt.

Wir können also in der Tat mit der Tastatur innerhalb

eines Programms Zahlenwerte in dasselbe eingeben, aber warum nur Zahlen?

Die Antwort ist einfach: Einer Variablen, die den Namen »A«, oder »ZAHL« oder »SUMME« hat, dürfen nur Zahlen zugeordnet werden.

Das heißt aber nicht, daß wir auf schriftliche Anweisungen, die aus Buchstaben, Zeichen oder ganzen Wörtern bestehen, verzichten müssen. Der INPUT-Befehl akzeptiert sie auch, aber nur, wenn wir String-Variable verwenden.

Wir können nämlich jetzt mit dem Befehl

INPUT A$

die Eingabe einer Information in Form eines Strings programmieren.

Bitte löschen Sie das letzte Programm mit NEW und geben Sie die beiden früheren INPUT-Programmzeilen neu ein und zwar jetzt so:

```
→ 20 INPUT A$
  30 PRINT A$
```

Wenn Sie nun RUN eingeben, wartet der Computer mit dem Fragezeichen, bis Sie einen String eingeben und mit RETURN abschließen. Dann erst druckt er den String aus. Der eingegebene String A$ darf maximal 88 Zeichen enthalten.

In einem benutzerfreundlichen Programm sollte natürlich bei dem wartenden Fragezeichen von INPUT dabeistehen, welche Art von Eingabe erwartet wird. Der Benutzer des Programms sieht ja schließlich die Programmzeile nicht, in welcher der INPUT-Befehl – mit oder ohne »$« – steht.

Eine derartige Angabe können wir leicht erzeugen, indem wir eine entsprechende PRINT-Zeile vor den INPUT-Befehl der Zeile 20 setzen:

```
→ 10 PRINT"TEXT-EINGABE"
  20 INPUT A$
  30 PRINT A$
```

Als Ergebnis erhalten wir auf dem Bildschirm das Wort »TEXT-Eingabe«, darunter das Fragezeichen und wie gehabt den wartenden Cursor.

Wem das Untereinander nicht gefällt, der kann mit dem Semikolon als String-Kleber alles auf eine Zeile bringen:

```
→ 10 PRINT"TEXT-EINGSABE";
```

Aber es geht noch viel eleganter !!

Der INPUT-Befehl selbst kann die Angabe enthalten. Wir nennen das einen »Kommentar« oder auf englisch ein »Prompt«.

```
→ 10
  20 INPUT"TEXT-EINGABE";A$
  30 PRINT A$
```

Zuerst wird die Zeile 10 gelöscht. Die Zeile 20 bringt dasselbe Ergebnis wie vorher die beiden Zeilen 10 und 20 zusammen.

Der Kommentar hinter dem INPUT muß immer zwischen Anführungszeichen stehen, darf maximal 38 Zeichen – auch Leerstellen zählen dazu – lang sein und muß von der Variablen (egal ob numerisch oder String) durch ein Semikolon getrennt sein.

Und noch eine feine Einrichtung hat der INPUT-Befehl: Man darf hinter ihn mehrere Variable hängen, die dann allerdings alle eingegeben werden müssen. Die Schreibweise ist der des PRINT-Befehls sehr ähnlich:

```
→ 40 INPUT"4 ZAHLEN";A,B,C,D
  50 PRINT A;B;C;D
  60 INPUT"3 STRINGS";A$,B$,C$
  70 PRINT A$ B$ C$
```

Sie sehen deutlich, daß der INPUT-Befehl ein Komma zur Trennung der Variablen verlangt, der PRINT-Befehl dagegen ein Semikolon, was, wie schon erwähnt, bei String-Variablen weggelassen werden kann.

Wenn Sie nun trotz des Kommentars, 4 Zahlen einzugeben, nur eine einzige eingeben und die RETURN-Taste drücken, gibt sich der Computer nicht zufrieden, sondern deutet mit einem doppelten Fragezeichen unmißverständlich darauf hin, daß noch weitere Eingaben erforderlich sind.

Die Zahlen, hintereinander eingegeben, müssen auch durch Kommata getrennt werden.

Geben Sie aber mehr Werte ein, als durch die Anzahl der Variablen hinter dem INPUT-Befehl verlangt werden, erscheint die Fehlermeldung »EXTRA IGNORED«.

Basic-Befehl Nr. 9 INPUT

- darf nur innerhalb eines Programms (also nur mit Zeilennummer) eingesetzt werden
- wird immer in der Schreibweise: INPUT (Variable) verwendet
- druckt ein Fragezeichen auf den Bildschirm und wartet auf eine Eingabe von der Tastatur, die mit RETURN abgeschlossen werden muß
- akzeptiert numerische Variable (Zahlen) und String-Variable (Text)
- weist eine falsche Zuordnung zurück, das heißt eine Zahl kann nicht für eine String-Variable und ein String nicht für eine normale Variable eingegeben werden
- mit einem einzigen INPUT-Befehl kann die Eingabe von mehreren Variablen abgefragt werden. Diese Variablen müssen durch Kommata voneinander getrennt sein
- werden nicht alle geforderten Variablen eingegeben, fragt INPUT mit einem doppelten Fragezeichen nach den fehlenden Werten
- werden mehr Werte als gefordert eingegeben, meldet der Computer dies mit einer Fehlermeldung »EXTRA IGNORED«
- bei String-Eingaben dürfen maximal 88 Zeichen eingegeben werden
- INPUT kann mit einem Kommentar versehen werden, den es vor dem Fragezeichen ausdruckt. Dieser Kommentar muß so geschrieben werden: INPUT "KOMMENTAR";(Variable)
- der Kommentar darf maximal 38 Zeichen lang sein

Diese Eingabemethode für Strings funktioniert gut, aber sie hat ein paar Eigenheiten und Einschränkungen: Dazu ein paar Beispiele:

Starten Sie die Zeilen 10 und 20 mit RUN. Wenn Sie hinter dem blinkenden Fragezeichen eingeben:

```
WERT:ZAHL  oder  WERT,ZAHL
```

erscheint eine Fehlermeldung und der Ausdruck:

EXTRA IGNORED
WERT

Der Text ab dem Komma beziehungsweise ab dem Doppelpunkt wird unterschlagen.

Die Eingabe:

```
"WERT" IST
```

führt zur Fehlermeldung REDO FROM START.

Bei einer Eingabe:

```
"DER WERT
```

wird das Anführungszeichen nicht geschrieben.

Dagegen wird die folgende Eingabe akzeptiert:

```
DER "WERT" IST
```

Sie sehen, man muß aufpassen.

FAZIT

1. Komma und Doppelpunkt dürfen im einzugebenden Text nicht vorkommen
2. einen Zeilensprung mit RETURN einzugeben geht nicht, da diese Taste den INPUT-Vorgang beendet
3. der Text darf nicht mit einem Anführungszeichen anfangen

4. der Text – mit Leerstellen – darf nicht länger als 88 Zeichen sein
5. der Eingabevorgang kann nicht mit der STOP-Taste abgebrochen werden.

Der INPUT-Befehl in Verbindung mit der IF-THEN-Abfrage ist eine wichtige Möglichkeit zur Steuerung und Beeinflussung von Programmen durch den Benutzer. Wir werden sie immer wieder antreffen. Zwei kleine Beispiele sollen diese Befehlskombination verdeutlichen:

In einem Spielprogramm soll der Spieler eine Zahl zwischen 5 und 10 eingeben. Das geht so:

```
→ 10 PRINT"GEBEN SIE EINE ZAHL ZWISCHEN
     5 UND 10 EIN"
  20 INPUT "ZAHL (5-10)";Z
  30 IF Z <  5 THEN 20
  40 IF Z > 10 THEN 20
  50 PRINT Z
```

Jede eingegebene Zahl Z, die kleiner als 5 oder größer als 10 ist, wird durch die Zeilen 30 und 40 durch Rücksprung auf den INPUT-Befehl zurückgewiesen.

Die Prüfvariable kann auch ein String sein. Bitte ergänzen Sie das Programm mit den folgenden Zeilen:

```
→ 60 INPUT "JA ODER NEIN";A$
  70 IF A$="JA" THEN GOTO 100
  80 IF A$="NEIN" THEN GOTO 110
  90 PRINT "FALSCHE EINGABE":GOTO 60
  100 PRINT "JA"
  110 PRINT "NEIN"
```

Das wollen wir jetzt zeilenweise analysieren.

Zeile 60 ist der INPUT-Befehl, der uns im Kommentar auffordert, entweder »JA« oder »NEIN« einzugeben.

Zeile 70 prüft, ob wir »JA« eingegeben haben. Ist das der Fall, dann springt sie auf Zeile 100, und wir erhalten den Ausdruck »JA«. Haben wir nicht das »JA« eingegeben, geht das Programm in der nächsten Zeile (80) weiter.

In Zeile 80 wird geprüft, ob wir »NEIN« eingegeben haben. Wenn das zutrifft, springt sie auf Zeile 110, und das Wort »NEIN« wird ausgedruckt. Haben wir aber »NEIN« auch nicht eingegeben, sondern irgend ein anderes Wort (String), dann geht das Programm in der nächsten Zeile (90) weiter.

Ich hoffe, Sie sehen wieder das typische Muster des IF..THEN-Befehls.

In Zeile 90 schließlich wird die Ermahnung ausgedruckt, doch nur mit »JA« oder »NEIN« zu antworten, und mit dem Rücksprung GOTO 60, mit dem Doppelpunkt als zweitem Befehl der Zeile 90 angehängt, wird eine neue Eingabe verlangt.

Einen kleinen Fehler hat das Programm noch. Wenn Sie es mit RUN von Anfang an oder mit RUN 60 nur ab der INPUT-Zeile starten und gleich mit »JA« antworten, kommt zuerst Zeile 100 zum Zuge, aber gleich danach auch Zeile 110, und wir erhalten beide Ausdrucke, »JA« und »NEIN«.

Zeile 110 können wir in diesem Fall ausblenden, entweder durch eine Überbrückung mit GOTO (irgendwohin) oder mit einem neuen BASIC-Befehl, der das Programm mit Zeile 100 beendet. Er heißt END.

Endstation mit END

Ändern Sie bitte Zeile 100 ab:

```
→ 100 PRINT"JA":END
```

Jetzt klappt es.

Der END-Befehl beendet also die Ausführung eines Programms.

Basic-Befehl Nr. 10 END

Dieser Befehl steht für sich allein oder nach einem IF..THEN-Befehl als Aktionsanweisung.

Er beendet sofort ein Programm und der Computer meldet sich mit READY und blinkendem Cursor.

9.2. Eingabe mit GET

In Basic gibt es noch einen zweiten Befehl, der im Prinzip das gleiche wie INPUT macht. Nur im Detail der Ausführung unterscheidet er sich vom Kollegen INPUT. Diese Unterschiede aber machen ihn zu einer wertvollen Alternative.

Der Befehl heißt

GET (Variable)

was soviel bedeutet, wie »holen, bekommen«.

Der GET-Befehl holt also einen Wert und weist ihn der hinter ihm stehenden Variablen zu. Diese Variable kann eine numerische oder eine String-Variable sein.

Unterschied zwischen GET und INPUT

- INPUT wartet, bis eine Eingabe mit der RETURN-Taste abgeschlossen ist
- GET wartet nicht, sondern prüft lediglich, ob eine Taste gedrückt worden ist.
- INPUT erlaubt die Eingabe bis zu 78 Zeichen
- GET holt immer nur 1 Zeichen
- INPUT meldet sich mit Fragezeichen und Cursor
- GET meldet sich auf dem Bildschirm nicht

Die Frage stellt sich, wie der GET-Befehl prüfen kann, ob eine Taste gedrückt worden ist, wenn er nicht wartet. Daß er gerade in dem kurzen Zeitpunkt prüft, in dem zufällig der Tastendruck stattfindet, ist mehr als unwahrscheinlich.

Und in der Tat, wenn nicht das Zeichen jeder gedrückten Taste in einen Pufferspeicher käme, ehe es weiter verwendet wird, hätte der GET-Befehl keine Chance, jemals eine Eingabe zu erwischen.

Der Tastaturpuffer

Diesen Speicher des Computers, in dem die Zeichen gedrückter Tasten erst einmal zwischengelagert werden, möchte ich Ihnen kurz zeigen.

Dazu brauchen wir eine Zählschleife, wie wir sie beim GOTO-Befehl erstmals angewendet haben. Hier brauche ich sie, um eine sogenannte Zeitverzögerungsschleife zu bauen. Diese tut das, was ihr Name sagt: sie zählt eine gewisse Zeit still vor sich hin und verzögert so den nächsten Programmschritt. Entfernen Sie bitte alle Programmzeilen mit NEW

```
→ NEW
  30 X=X+1
  40 IF X <> 500 THEN 30
```

Mit RUN gestartet, läuft die Schleife in diesen beiden Zeilen 500mal durch, bis sich der Editor mit READY und Cursor wieder meldet.

Diese Zeit nützen wir, um so viele Tasten wie möglich hintereinander zu drücken. Natürlich sehen wir nichts auf dem Bildschirm, während das Programm der Zeitschleife läuft. Die Behauptung, daß die gedrückten Zeichen in einen Pufferspeicher kommen, bewahrheitet sich nach dem Ende der Zeitschleife.

Nach dem READY holt nämlich der Editor alle Zeichen aus dem Tastaturpuffer und druckt sie aus. Da dieser Puffer nur 10 Zeichen speichern kann, sehen Sie nur 10 Zeichen, selbst wenn Sie es geschafft haben, mehr einzugeben.

Also nochmal:

→ RUN eingeben
möglichst viele Tasten drücken, aber nicht gleichzeitig nach dem READY die Zeichen zählen

In diesem Tastaturpuffer schaut also der GET-Befehl nach, ob vorher ein Tastendruck ein Zeichen darin untergebracht hat. Wenn ja, dann verwendet er es, falls nein, kommt die nächste Programmzeile dran.

Um das zu zeigen, erweitern wir das obige Programm um vier weitere Zeilen:

```
→ 10 GET A
  20 PRINT A
  30 X=X+1
  40 IF X <> 100 THEN 30
  50 X=0
  60 GOTO 10
```

In Zeile 10 schaut der GET-Befehl im Tastaturpuffer nach einem Zeichen. Was immer er findet, druckt Zeile 20 auf den Bildschirm. Eine 0 signalisiert, daß der Puffer leer war.

Die Zeilen 30 und 40 zählen von 1 bis 100, wie beim Versteckspielen, und machen dann nach 100 Zähleinheiten in Zeile 50 weiter, in der die Zählvariable X wieder auf Null gesetzt wird.

Zeile 60 springt zurück auf den GET-Befehl und alles fängt mit X=0 wieder von vorn an. Das Resultat ist eine langsam fortschreitende senkrechte Kolonne von Ziffern links am Bildschirm. Die Geschwindigkeit ist durch die Prüfzahl in Zeile 40 einstellbar.

Während der Zählschleife haben wir nun Gelegenheit, Zahlen einzugeben, die als einzelne Ziffern vom GET-Befehl geholt und durch Zeile 20 ausgedruckt werden.

Die Zeitschleife der beiden Zeilen 30 und 40 können wir natürlich auch mit der FOR-TO-NEXT-Schleife schreiben. Da diese Schleifenart aber viel schneller abläuft, muß die Zählvariable 10mal so groß sein.

```
→ 30 FOR X=0 TO 1000: NEXT
  40  (entfällt)
  50  (entfällt)
```

Wenn Sie einen Buchstaben eingeben, bricht das Programm mit einer Fehlermeldung ab.

Um Buchstaben und Zeichen eingeben zu können, müssen wir die numerische Variable A in eine String-Variable A$ umwandeln – natürlich in beiden Zeilen 10 und 20.

Jetzt holt der GET-Befehl Buchstaben – aber auch Ziffern! Diese werden jedoch als Zeichen (einstellige Strings) und nicht als Zahlenwerte behandelt.

Basic-Befehl Nr. 11 GET
- der Befehl GET (Variable) holt eine Zahl oder ein Zeichen aus dem Tastaturpuffer und weist es der Variablen zu
- die Variable kann eine numerische oder eine String-Variable sein
- entspricht das Zeichen bzw. die Zahl nicht dem Variablentyp, wird mit Fehlermeldung abgebrochen
- GET wartet nicht. Falls der Puffer leer ist, weist er der Variablen den Wert 0 beziehungsweise einen sogenannten »Nullstring« (kein Leerzeichen, einfach gar nichts) zu
- im Gegensatz zu INPUT sind bei der Eingabe mit GET alle Zeichen erlaubt, auch Komma, Doppelpunkt und Semikolon

Jetzt will ich Ihnen eine echte Anwendung des GET-Befehls zeigen. Wir haben in Lektion 9 bei der INPUT-Eingabe einen Programmteil geschrieben, um den Benutzer zu fragen, ob er mit JA oder NEIN antworten will.

Mit dem INPUT-Befehl ging das so:

```
→ 10 PRINT"BEGINN"
  .
  .   (Programm)
  .
  320 INPUT"NOCH EINMAL (J/N)";A$
  340 IF A$="J" THEN 10
  350 IF A$ <> " N" THEN 320
  360 PRINT"AUF WIEDERSEHEN"
```

Ab Zeile 10 beginnt das Programm.
Zeile 320 fragt nach J(a) oder N(ein).
Zeile 340 prüft, ob die Eingabe "J(a)" ist, im Fall »wahr« springt das Programm an den Anfang zurück.

Im Fall »falsch« kommt Zeile 350 an die Reihe mit der Prüfung, ob die Eingabe ein ungültiges Zeichen ist – alle Zeichen außer »NEIN« sind ungültig. Leider ist die Eingabe »NEIN« auch falsch, denn wir prüfen ja nur auf »N«. Das ist ein Nachteil des INPUT-Befehls.

Im Fall »wahr« beziehungsweise »J« wird der INPUT-Befehl wiederholt. Wenn es ein »N« war, verabschiedet sich das Programm in Zeile 360.

Dasselbe machen wir jetzt mit dem GET-Befehl. Die Zeilen 10, 340, 350 und 360 bleiben unverändert.

```
→ 10 PRINT"BEGINN"
  .
  .   (Programm)
  .
  320 PRINT"NOCH EINMAL (J/N)?"
  330 GET A$:IF A$=""THEN 330
  340 IF A$="J"THEN 10
  350 IF A$ <>"N" THEN 320
  360 PRINT"AUF WIEDERSEHEN"
```

Zeile 320 druckt die Anfrage aus, die oben hinter dem INPUT-Befehl steht. In Zeile 330 schaut der GET-Befehl im Tastaturpuffer nach, ob er ein Zeichen enthält. Mit IF A$= " " prüft er, ob der Puffer leer ist. Die doppelten Anführungszeichen ohne Zwischenraum repräsentieren den bei der Befehlsbeschreibung genannten »Nullstring«, das heißt, im Puffer war kein Zeichen. Deswegen springt diese Zeile 330 auf sich selbst zurück und verharrt in dieser Schleife so lange, bis die Prüfbedingung des Nullstrings nicht mehr erfüllt ist. Das ist sie, sobald irgendeine Taste gedrückt worden ist. Die restlichen Zeilen prüfen genauso wie im Beispiel vorher. Der Hauptunterschied ist der, daß der GET-Befehl auch »NEIN« akzeptiert, prüft er doch immer nur 1 Zeichen, nämlich das erste – er würde natürlich auch »NAME« akzeptieren. Außerdem reagiert er sofort und nicht erst nach dem Drücken der RETURN-Taste. Prinzipiell möchte ich sagen, daß der GET-Befehl »intelligenter« ist als INPUT. Er bietet mehr Flexibilität und auch mehr Eleganz – eine Eigenschaft, in der sich oft gute und schlechte Programme unterscheiden.

FAZIT
1. der GET-Befehl wartet nicht
2. wenn er auf eine Eingabe warten soll, muß er mit einer »Nullstring-Abfrage« in eine einzeilige Warteschleife gelegt werden:

```
20 GET A$:IF A$="" THEN 20
```

Es könnte sein, daß der eine oder andere Leser jetzt seufzt und sich fragt, wie denn mit allen diesen Befehlen jemals ein Programm zusammenstellbar sei.

Ich habe in der Tat vor, so schnell wie möglich mit Ihnen ein erstes sinnvolles und für Sie reizvolles Programm zu entwickeln. Natürlich wird es ein Spiel sein. Wir brauchen dazu aber noch zwei weitere Befehle.

LEKTION 10: Zufallszahlen und ganze Zahlen

Der Zufall im Computer ist eigentlich ein Widerspruch in sich, da doch im Computer alles durch die Anweisungen des Programms fest vorgegeben ist. Und doch gibt es etwas Zufallähnliches.

Es ist sowohl bei mathematischen Anwendungen als auch bei Spielen oft erforderlich, von einer nicht vorher bekannten, völlig zufällig entstandenen Zahl auszugehen. Wie könnte sie anders heißen als »Zufallszahl«.

BASIC hat einen Befehl, der eine derartige Zufallszahl erzeugt. Er heißt:

RND (X)

Der Name ist abgeleitet aus *Random*, was auf deutsch »zufällig« bedeutet.

Das X, das in Klammern hinter dem Befehlswort steht, wird Argument genannt. Das X kann drei Werte haben:
- eine positive Zahl (egal, welcher Wert)
- eine negative Zahl
- die Zahl 0

Bevor wir darauf eingehen, möchte ich Ihnen die Wirkungsweise von RND zeigen, wie immer am besten durch ein Experiment.

```
→ 10 A=RND(1)
  20 PRINT A
  30 GOTO 10
```

Diese drei Zeilen bilden eine Schleife, die Ihnen in einem Zahlenstreifen die Werte zeigt, die RND(1) erzeugt. Wir sehen vielstellige Zahlenwerte, die abgerundet zwischen 0 und 1 liegen.

Ab und zu taucht in der Zahlenkolonne eine Zahl in eigenartiger Schreibweise auf, mit einer Ziffer vor dem Dezimalpunkt und einem »E–« gefolgt von einer Zahl. Das ist die sogenannte wissenschaftliche Schreibweise, die im Handbuch von Commodore genauer erklärt wird. Hier bitte ich Sie, diese Zahlen ganz einfach zu ignorieren.

Zurück zu den Zahlen, die von RND erzeugt werden.

Sie werden von einer Anfangszahl ausgehend durch eine komplizierte mathematische Formel errechnet, die zwar keine absolute Zufälligkeit garantiert, ihr aber sehr nahekommt.

Sie können sich vorstellen, daß die Zufälligkeit ganz wesentlich von der oben genannten Anfangszahl abhängt. Und gerade sie wird durch das Argument (X) beziehungsweise durch die drei Arten des Arguments bestimmt.

Das positive X beginnt nach dem Einschalten des Computers immer mit derselben Anfangszahl. Bei X=0 wird als Anfangszahl der Stand der inneren Computeruhr genommen. Bei einer negativen Zahl bildet diese selbst den Anfangswert.

Falls Sie mehr über diese Methode wissen wollen, finden Sie eine Erklärung im der 64'er Ausgabe 8/85 auf Seite 131.

Ich empfehle Ihnen, den Wert 0 zu nehmen. Mit 0 als Argument von RND will ich die drei Zeilen von oben neu schreiben, diesmal als sogenannten »Einzeiler«. Darunter versteht man ein Programm, welches in eine einzige Zeile paßt. Glauben Sie mir, in dieser Art sind schon ganze Kunstwerke veröffentlicht worden.

Diesen Anspruch erhebe ich nicht, nur den der Vereinfachung.

```
→ 10 PRINT RND(0):GOTO 10
```

Die Zeile braucht unbedingt eine Zeilennummer, damit der GOTO-Befehl eine Schleife bilden kann.

Basic-Befehl Nr. 12 RND(X)
- wird für X die Zahl 0 oder irgendeine positive Zahl genommen, erzeugt dieser Befehl eine zirka achtstellige Zahl zwischen Null und Eins. Ihr jeweiliger Wert ist sozusagen zufällig.

Zufallszahlen haben wir jetzt, aber leider in der Form eines Dezimalbruches. Wenn wir in einem Programm zum Beispiel würfeln wollen, brauchen wir Zufallszahlen von 1 bis 6. Dezimalbrüche helfen uns da nicht viel. Wir brauchen also eine Möglichkeit, diese Dezimalbrüche in ganze Zahlen umzuwandeln.

Auch dafür hat BASIC einen Befehl. Er lautet:

INT

Das ist die Abkürzung von *Integer*, was auf deutsch »ganze Zahl« bedeutet.

Der Dezimalbruch, der mit INT in eine ganze Zahl verwandelt werden soll, wird in Klammern hinter den Befehl geschrieben. Der Befehl ist im Direktmodus verwendbar:

```
→ PRINT INT(25.38)
```

Wir erhalten die Zahl 25.

INT macht es sich also leicht – es schneidet einfach alle Zahlen hinter dem Dezimalpunkt weg.

Es ist auch erlaubt, genau wie beim PRINT-Befehl eine Formel mit INT zu versehen:

```
→ PRINT INT(2.8908*567.8)
```

Das Ergebnis ist 1641.

Basic-Befehl Nr. 13 INT(A)
- wandelt einen Dezimalbruch A in eine ganze Zahl um
- A kann nicht nur ein Dezimalbruch, sondern eine mathematische Formel oder ein komplizierter mathematischer Ausdruck sein
- die Klammern dürfen nicht weggelassen werden

Diesen Befehl INT wenden wir jetzt an, um aus den unerwünschten Dezimalbrüchen des RND-Befehls ganze Zahlen zu machen.

Zuerst nehme ich der Klarheit halber die 3zeilige Version, welche die Zufallszahl einer Variablen zuordnet:

```
→ 10 A=RND(0)
  20 PRINT INT(A)
  30 GOTO 10
```

Der Unterschied zu vorher liegt hier in der Zeile 20. Das Resultat nach RUN ist enttäuschend, aber verständlich, denn der INT-Befehl macht aus Dezimalbrüchen, die kleiner als 0 sind, nur eine 0.

Zeile 20 muß verbessert werden, indem der Dezimalpunkt von A nach rechts verschoben wird. Das erreichen wir durch die Multiplikation mit 10, 100, 1000 und so weiter.

```
→ 20 PRINT INT (A*10)
```

liefert Zufallszahlen von 0 bis 9. Den Bereich 0 bis 99 erhalten wir durch:

```
→ 20 PRINT INT (A*100)
```

Als Einzeiler sieht diese letzte Version so aus:

```
→ 10 PRINT INT(RND(0)*100):GOTO 10
```

Beachten Sie bitte die Verschachtelung der Klammern:

INT(RND(0))

1. Klammer auf — 1. Klammer zu
2. Klammer auf — 2. Klammer zu

Wenn eine Klammer fehlt oder falsch ist, erhalten wir die Fehlermeldung »SYNTAX ERROR«. Am besten ist es, Sie überprüfen, daß die Anzahl der geöffneten und geschlossenen Klammern immer gleich ist.

Im Fazit sehen Sie ein Kochrezept, mit dem Sie Zufallszahlen innerhalb des von Ihnen gewünschten Bereiches erzeugen können.

FAZIT

Die Formel:

```
→ A = INT(RND(0)*X)+Y
```

erzeugt ganze Zahlen innerhalb des Zahlenbereiches Y bis Y+X–1.

Beispiel:

Zufallszahlen von 20 bis 90 machen Y zu 20 und erfordern ein X, das aus der Formel Y+X–1=90 errechnet wird. Anders geschrieben lautet diese Formel X=90–Y+1. Das ergibt in unserem Fall X=71.

```
→ 10 PRINT INT(RND(0)*71)+20
```

Im Zweifelsfall lohnt es sich immer, die Formel mit einer derartigen Zeile 10 auszuprobieren.

LEKTION 11: Ein vollständiges Programm

Wir sind gerüstet für ein erstes Programm. Es heißt »ZAHLEN RATEN« und ist aus dem Buch »Computerspiele und Knobeleien« von Rüdiger Baumann abgeleitet.

Aufgabe:

Eine vom Computer zufällig gewählte Zahl zwischen 1 und 100 soll in möglichst wenigen Versuchen erraten werden. Nach jedem Rateversuch gibt der Computer einen Hinweis »zu groß« oder »zu klein«. Wurde richtig geraten, nennt der Computer die Zahl der Versuche und stellt anheim, das Spiel noch einmal zu versuchen oder zu beenden.

11.1. Die Planung eines Programms

Ich habe ganz am Anfang dieses Kurses erwähnt, daß Basic dazu verleitet zu »hacken«. In der Tat führt die an sich sehr positive Eigenschaft dieser Programmiersprache, nämlich schon kleinste Bruchstücke eines Programms eingeben und dann gleich ausprobieren zu können, leicht zu einem sogenannten »Spaghetti-Programm«. Die einzelnen Bruchstücke kann man nämlich stehenlassen und die anderen Teile irgendwie dazwischenschieben oder mit GOTO irgendwo anhängen. Das Resultat ist aber am Schluß ein wilder Haufen von Befehlszeilen, die zwar funktionieren, aber nicht mehr nachvollziehbar sind.

Es ist daher anzuraten, sich den Ablauf eines Programms vorher zu überlegen, was ja nicht schwer ist, weil ein Computer nie mehrere Sachen gleichzeitig macht, sondern alle streng hintereinander abwickelt. So einen Ablauf kann man schön aufs Papier malen, und das will ich Ihnen jetzt zeigen.

Zuerst schreiben wir ganz einfach alle Vorgänge der Reihe nach auf. Wenn die Reihenfolge uns nicht paßt, wird sie einfach abgeändert.

Am Ende kann es so aussehen:

1. Bildschirm löschen
2. Überschrift schreiben
3. Zufallszahl zwischen 1 und 100 erzeugen
4. Zähler für Zahl der Versuche auf Null stellen
5. Anweisungen an den Spieler
6. Eingabe einer geratenen Zahl
7. Zähler der Versuche um 1 erhöhen
8. Prüfung der geratenen Zahl auf:
   - zu groß: Hinweis geben und neuer Versuch
   - zu klein: Hinweis geben und neuer Versuch
   - richtig: Zahl der Versuche ausdrucken
9. Frage ob noch einmal
   - ja: zurück zum Anfang
   - nein: Verabschiedung, Ende

Alles was jetzt noch zu tun bleibt, ist, die einzelnen Punkte der Reihe nach mit den Befehlen, die wir kennen, als Programmzeilen zu schreiben.

Übrigens: Ich schreibe zuerst nur die »aktiven« Programmzeilen. Alle anderen Teile des Programms, die nur der Lesbarkeit dienen, folgen später. Dadurch erklären sich auch die Sprünge in den Zeilennummern, die ich Sie bitte zu beachten.

11.2. Die Durchführung

Punkt 1 und 2:

Den Bildschirm löschen wir mit einem PRINT-Befehl, hinter dem wir nach dem ersten Anführungszeichen die geSHIFTete CLR/HOME-Taste drücken – was bekanntlich im Direktmodus den Löschvorgang auslöst. Im Listing erscheint dann ein invertiertes Herz. Was es damit auf sich hat, erkläre ich im Anschluß an das Programm.

```
→ 10 PRINT" {SHIFT CLR/HOME}"
  20 PRINT"****** ZAHLEN RATEN ******"
```

Punkt 3:

Der Bereich der Zufallszahl soll zwischen 1 und 100 liegen. Entsprechend dem dafür angegebenen Kochrezept INT(RND(0)*X)+Y errechnen wir wieder mit der unteren Grenze 1=Y und der oberen Grenze X+Y-1=100 das uns noch fehlende X:

X+1-1=100

X=100

Dann ergibt sich für die Zufallszahl »Z« die folgende Programmzeile:

```
→ 70 Z=INT(RND(0)*100)+1
```

Punkt 4:

Der Zähl-Variablen für die Anzahl der Versuche geben wir den Namen »V«. In Zeile 80 wird sie auf Null gesetzt.

```
→ 80 V=0
```

Punkt 5:

Die Anweisungen an den Spieler stehen in Zeile 120 und 130. Sie können sie in Listing 1 – so nennen wir den kompletten Ausdruck eines Programms – in allen Einzelheiten nachlesen.

Sie können die Anweisung aber Ihrem Geschmack entsprechend auch anders schreiben oder anders anordnen.

Punkt 6:

Die Aufforderung zur Eingabe einer Zahl und die Eingabe selbst besorgt uns wieder der INPUT-Befehl

```
→ 140 PRINT
  150 INPUT"WELCHE ZAHL IST ES";R
```

Vor dem INPUT-Befehl drucken wir erst eine Leerzeile aus, damit es besser aussieht.

Der einzugebenden Zahl stellen wir die numerische Variable »R« zur Verfügung, diesmal ohne $-Zeichen, da wir ja Zahlen eingeben wollen.

Punkt 7:

Sobald die Zahl eingegeben ist, muß die Variable V des Versuche-Zählers um 1 erhöht werden.

```
→ 160 V=V+1
```

Punkt 8:

Jetzt kommt die Prüfung mit IF THEN. Ich habe sie wegen der nachfolgenden Hinweise ZU GROSS, ZU KLEIN und RICHTIG in 3 Gruppen zu je 2 Stufen aufgebaut.

Das Grundprinzip dieser Prüfung liegt im Vergleich der beiden Zahlen, nämlich der geheimen Zahl des Rechners »Z« und der eingegebenen Zahl »R« des Raters. Zeile 200 prüft auf »größer«, Zeile 210 auf »kleiner«, Zeile 220 auf »gleich«.

```
→ 200 IF R < Z THEN PRINT R;:GOTO xxx
  xxx PRINT"IST ZU GROSS":GOTO 140
```

Die Zeilennummer xxx tragen wir dann nach, sobald klar ist, wieviele Zeilen noch dazwischenliegen.

Wichtig in Zeile 200 ist das Semikolon nach der geratenen Zahl R. Es klebt wieder einmal den nachfolgenden Satz der Zeile xxx direkt hinter den Wert der Zahl R, so daß auf dem Bildschirm der Satz erscheint:

»(Zahl R) IST ZU GROSS«.

Genauso ist es mit den beiden anderen Prüfungen:

```
→ 210 IF R > Z THEN PRINT R;:GOTO yyy
  yyy PRINT"IST ZU KLEIN":GOTO 140
  220 IF R = Z THEN PRINT R;:GOTO zzz
  zzz PRINT"IST RICHTIG"
```

Aus der Reihenfolge wird jetzt klar, daß xxx die Zeile 230 ist, entsprechend yyy=240 und zzz=250. Bitte tragen Sie dies nach.

Die beiden falschen Ergebnisse in Zeile 230 und 240 füh-

ren also zum Rücksprung auf eine neue Eingabe, die ab Zeile 140 beginnt.

Ein richtiges Ergebnis führt in die nächsthöhere Zeile 260 weiter.

In Zeile 260 und 270 drucken wir jetzt den letzten Stand des Versuche-Zählers V aus, eingebettet in Text, von dem er durch zwei Semikolons getrennt werden muß:

```
→ 260 PRINT"SIE HABEN";V;"VERSUCHE ";
  270 PRINT"BENOETIGT"
```

Ich habe mit Absicht den Text, der ja leicht in eine Zeile gepaßt hätte, in zwei Zeilen getrennt. Erstens verbessert er das Format des Listings. Zweitens aber wollte ich Ihnen zeigen, wie man trotzdem den Text in eine Zeile ausgedruckt erhält. Das Geheimnis liegt wieder im Semikolon nach dem letzten Wort der Zeile 260; zusätzlich aber ist die Leerstelle vor dem abschließenden Anführungszeichen wichtig. Ohne sie würden die beiden Wörter »VERSUCHE« und »BENOETIGT« zu einem einzigen Wort verklebt werden.

Punkt 9:

Es bleibt noch die Frage nach einer Wiederholung. Das machen wir mit dem GET-Befehl.

Zeile 320 stellt die Frage, in Zeile 330 wartet der GET-Befehl mit einer Nullstring-Schleife auf eine gedrückte Taste. Zeile 340 prüft auf »J(a)« und springt zurück auf den Anfang, Zeile 350 prüft auf ungültige Eingaben mit Wiederholung der Aufforderung, und wenn »N(ein)« eingegeben ist, schließt Zeile 360 das Programm mit höflicher Verabschiedung.

```
→ 320 PRINT"NOCH EINMAL (J/N) ?"
  330 GET A$:IF A$="" THEN 330
  340 IF A$="J"THEN 10
  350 IF A$ <> "N" THEN 320
  360 PRINT:PRINT:PRINT"AUF WIEDERSEHEN"
  370 END
```

So, das war doch nicht schwer, oder?

Alles, was passieren kann, sind Tippfehler beim Eingeben.

Im nachfolgenden Listing 1 ist das ganze Programm abgedruckt.

Zwei Dinge fallen dabei auf:

- Am rechten Rand stehen Zahlen in eckigen Klammern. Es sind Prüfzahlen, die dann entstehen, wenn das Programm mit der Eingabehilfe »Checksummer 64 V3« eingegeben wird, welches Sie in jeder Ausgabe der 64'er finden.
- Viele Zeilen sind bisher in der Beschreibung des Programms nicht vorgekommen, zum Beispiel die Zeilen 40, 50, 60, 90, 100 etc.

Sie dienen der besseren Lesbarkeit des Listings. Sie sind eine eigene kurze Lektion wert.

LEKTION 12: Lesbarkeit und Struktur eines Programms

Eine Methode, das Listing lesbar anzulegen, besteht darin, Leerzeichen an entsprechenden Stellen einzugeben. Basic macht nämlich keinen Unterschied zwischen:

PRINT A$; B$ und PRINTA$;B$.

Die erste Version braucht zwar 3 Speicherplätze mehr – eben die 3 Leerzeichen, aber Version 2 ist schlecht lesbar.

Zur besseren Lesbarkeit gehört auch, daß ab und zu eine Leerzeile eingefügt wird, die den Programmablauf nicht beeinflußt. Im Programm sind dies die Zeilen 30, 270 und 290. Man erreicht dies durch einen Doppelpunkt nach der Zeilennummer, ohne Befehl.

Eine dritte Art, das Listing zu verbessern, ist der Befehl REM

Er kommt von Remark und bedeutet »Bemerkung«. Mit diesem Befehl kann man Text in das Listing eingeben, der aber vom Programm völlig ignoriert wird. Im Beispiel sind dies die Zeilen 40 und 380.

Basic-Befehl Nr. 14 REM
Dieser Befehl erlaubt, einen Text oder Zeichen – mit Zeilennummer versehen – in den Ausdruck eines Programms (Listing) einzufügen, ohne daß er im Ablauf des Programms erscheint. Es werden dabei keine Anführungszeichen verwendet.
Dieses Verfahren erhöht die Lesbarkeit des Listings.

Der vierte Punkt, der Ihnen im nebenstehenden Listing des Programms Nr. 1 auffallen müßte, ist die Tatsache, daß bei allen PRINT-Befehlen der Text auf der linken Seite zwar mit einem Anführungszeichen anfängt, aber rechts nicht mit dem zweiten Anführungszeichen aufhört.

Das ist am Ende einer Zeile erlaubt, weil ja die Zeile mit der RETURN-Taste abgeschlossen worden ist.

FAZIT
1. Leerzeilen, die nur im Listing auftreten, das Programm aber nicht beeinflussen, erzeugt man durch einen Doppelpunkt hinter der Zeilennummer
2. Bei der Abarbeitung einer Programmzeile ignoriert Basic alle Leerzeichen, solange sie nicht zwischen Anführungszeichen stehend zu einem String gehören. Sie können daher weggelassen werden.
3. Am Ende einer Zeile kann das abschließende Anführungszeichen weggelassen werden.

Der fünfte Punkt betrifft die einzelnen Blöcke des Programms. In Listing 1 – und natürlich schon bei der Planung des Programms – habe ich einzelne Gruppen gebildet, die dem Programm eine logische und übersichtliche Struktur geben.

Diese Methode nennt man »Strukturiertes Programmieren«, was besonders durch die sogenannten »Unterprogramme« ermöglicht wird. Diese Unterprogramme kommen erst in Lektion 24 an die Reihe.

LEKTION 13: Programmierte Steuertasten und der ASCII-Code

In Zeile 10 des Programms »Zahlen Raten« kommt ein invertiertes Zeichen in Anführungszeichen vor. Es ist das Symbol der Steuertaste CLR, eingebettet in ein Programm. Was bedeutet das?

Der Computer nützt hier eine Eigenschaft des PRINT-Befehls aus. PRINT druckt bekanntlich sowohl im Direktals auch im Programm-Modus alle Zeichen auf den Bildschirm, die in Anführungszeichen hinter ihm stehen.

```
→ PRINT"AAABBB"
```

druckt nach RETURN diese 6 Buchstaben aus. Was ist aber mit den Steuertasten CLR, HOME INST, DEL CRSR rauf/runter/links/rechts, und was ist mit den Farbtasten?

Versuchen Sie es! Sie werden sehen, daß auch diese Tasten, zwischen Anführungszeichen, ein Symbol auf dem Bildschirm erzeugen, und zwar immer invertiert. In einem PRINT-Befehl erscheinen diese invertierten Zeichen nur im Listing.

Bei der Ausführung des PRINT-Befehls wird ihre Funktion ausgeführt !!

Das müssen Sie ausprobieren. Geben Sie einen PRINT-Befehl mit Buchstaben, gefolgt von der CTRL-Taste, gleichzeitig gedrückt mit der 8-Taste (YEL) und dann wieder Buchstaben ein.

```
10 PRINT"{CLR}                                      <186>
20 PRINT"******** ZAHLEN RATEN *******          <180>
30 PRINT                                        <132>
40 :                                            <016>
50 REM+++++++ VORBEREITUNG ++++++++++++         <244>
60 :                                            <036>
70 Z=INT(RND(0)*100)+1                          <060>
80 V=0                                          <065>
90 :                                            <066>
100 REM+++++++ RATEN +++++++++++++++++          <082>
110 :                                           <086>
120 PRINT"ICH HABE MIR EINE ZAHL                <144>
130 PRINT"ZWISCHEN 1 UND 100 GEMERKT.           <252>
140 PRINT                                       <242>
150 INPUT"WELCHE ZAHL IST ES ";R                <020>
160 V=V+1                                       <148>
170 :                                           <146>
180 REM+++++++ PRUEFEN +++++++++++++++          <096>
190 :                                           <166>
200 IF R>Z THEN PRINT R;:GOTO 230               <052>
210 IF R<Z THEN PRINT R;:GOTO 240               <126>
220 IF R=Z THEN PRINT R;:GOTO 250               <152>
230 PRINT"IST ZU GROSS":GOTO 140                <195>
240 PRINT"IST ZU KLEIN":GOTO 140                <043>
250 PRINT"IST RICHTIG"                          <067>
260 PRINT"SIE HABEN";V;"VERSUCHE ";             <020>
270 PRINT"BENOETIGT                             <030>
280 :                                           <002>
290 REM+++++ WIEDERHOLUNG ODER ENDE ++++        <153>
300 :                                           <022>
310 PRINT:PRINT:PRINT                           <181>
320 PRINT"NOCH EINMAL (J/N) ?                   <040>
330 GET A$: IF A$=""THEN 330                    <205>
340 IF A$="J"THEN 10                            <137>
350 IF A$<>"N" THEN 320                         <191>
360 PRINT:PRINT:PRINT"AUF WIEDERSEHEN           <017>
370 END                                         <118>
© 64'er
```

Listing 1. Das erste Programm – einfaches Zahlenraten

→ PRINT"AAA {CTRL 8} BBB"

Statt (CTRL 8) sehen Sie auf dem Bildschirm ein invertiertes MTC.

Sobald Sie jetzt <RETURN> drücken, werden die B und alles folgende in Gelb gedruckt. Wenn Sie die ursprüngliche Farbe wiederherstellen wollen, drücken Sie entweder auf die CBM-Taste und die 7 gleichzeitig, oder aber auf STOP und RESTORE.

Genauso wie mit den Farb-Tasten geht es mit den Cursortasten. Zur Abwechslung machen wir das im Programm-Modus.

→ 10 PRINT"ZZZ {4mal CRSR-rechts} XXX"

Nach RUN werden die Z noch normal, die X aber um 4 Stellen nach rechts versetzt gedruckt.

Die folgende Tabelle zeigt Ihnen die invertierten Steuerzeichen zur leichteren Identifzierung mit der Funktionsbeschreibung.

(Die letzten acht Steuerfunktionen können nicht mit Tasten direkt erzeugt werden. Sie müssen separat als reverse Zeichen eingetippt werden.)

Es gibt dabei einige Funktionen, die ihre eigenen invertierten Steuerzeichen, aber keine eigenen Tasten dafür haben bzw. die nicht mit den Tasten erzeugt werden können.

Funktion	Taste(n)	Zeichen	ASCII
schwarz	<CTRL 1>		144
weiß	<CTRL 2>		5
rot	<CTRL 3>		28
lila	<CTRL 4>		159
purpur	<CTRL 5>		156
grün	<CTRL 6>		30
blau	<CTRL 7>		31
gelb	<CTRL 8>		158
orange	<CBM 1>		129
braun	<CBM 2>		149
hellrot	<CBM 3>		150
hellgrau	<CBM 4>		151
mittelgrau	<CBM 5>		152
hellgrün	<CBM 6>		153
hellblau	<CBM 7>		154
dunkelgrau	<CBM 8>		155
Revers ein	<CTRL 9>		18
Revers aus	<CTRL 0>		146
Cursor rauf	<SHIFT CRSR>		145
Cursor ab	<CRSR>		17
Cursor links	<SHIFT CRSR>		157
Cursor rechts	<CRSR>		29
Schirm löschen	<SHIFT CLR/HOME>		147
Cursor Home	<CLR/HOME>		19
Klein-/Groß-buchstaben	<SHIFT CBM>		14
Großbuch-staben/Zeichen	<SHIFT CBM>		142
Funktionstaste F1	<F1>		133
F2	<SHIFT F1>		137
F3	<F3>		134
F4	<SHIFT F3>		138
F5	<F5>		135
F6	<SHIFT F6>		139
F7	<F7>		136
F8	<SHIFT F7>		140
Insert	<SHIFT INST/DEL>		148
Delete	–		20
Return	–		13
Shift-Return	–		141
Lock (Sperre der Zeichen-satzumschaltung)	–		8
Unlock (Sperre aufheben)	–		9
1. Zeichensatz	–		142
2. Zeichensatz	–		14

Ihre Funktion kann nur dadurch erzielt werden, daß beim Eintippen für sie zuerst eine Leerstelle freigelassen wird. Nach <RETURN> muß man dann mit dem Cursor auf diese Leerstelle fahren, dann REVERS-ON (mit der CTRL-Taste) drücken und anschließend das in der Tabelle dargestellte (reverse) Zeichen eingeben. Diese Aktion wird wieder mit <RETURN> abgeschlossen.

Ganz rechts in der Tabelle der Steuertasten ist eine Spalte von Zahlen, die unter der Überschrift ASCII stehen. Was dahintersteckt, sei kurz beschrieben.

Alle Computer verwenden intern irgendwelche Code-Zahlen, um die Zeichen, Buchstaben und Zahlen im Rechenwerk, im Speicher und in den anderen Einheiten des Computers darzustellen.

Theoretisch kann das jeder Computerhersteller machen, wie er will. Nur, wenn Daten von einem Gerät an ein anderes geliefert werden, zum Beispiel vom Computer an einen Drucker, müssen die Daten einem international standardisierten Code entsprechen. Dieser Standard heißt »American Standard Code for Information Interchange«, abgekürzt ASCII.

Jedes Zeichen, jede Zahl und jede Funktion hat seinen eigenen Code-Wert. Sie sind (fast) alle in einer Tabelle im Anhang F des Commodore-Handbuches aufgelistet. In der Tabelle sind ganz einfach diese Werte nur für die Steuertasten ausgewählt und dargestellt.

Wir können aber ganz leicht ein kleines Programm schreiben, das uns die Abfrage aller ASCII-Codes gestattet. Dazu muß ich allerdings einen Befehl vorwegnehmen, den ich in der nächsten Lektion erkläre. Aber das macht nichts - Sie können mir sicher folgen.

```
→ 10 INPUT A$
  20 A=ASC(A$)
  30 PRINT A
  40 GOTO 10
```

Der Pfiff liegt natürlich im neuen Befehl ASC(A$) in Zeile 20. Er wandelt den ersten Buchstaben des per INPUT eingegebenen und mit <RETURN> abgeschlossenen Strings in seinen ASCII-Codewert um, den wir in Zeile 30 ausdrucken.

Zeile 40 verschafft dem Programm die Wiederholbarkeit, die bei INPUT nur durch die STOP- und RESTORE-Taste aufgehoben werden kann. Jetzt können Sie nach Herzenslust alle Tasten und Tastenkombinationen nach ihrem ASCII-Code abfragen und so die unvollständige Liste im Commodore-Handbuch vervollständigen.

LEKTION 14: String-Befehle

Strings - auf deutsch »Zeichenketten« - haben wir ausführlich in Lektion 3 und Lektion 6 behandelt. Basic kennt mehrere Befehle, mit denen Strings verändert, verglichen und sonst irgendwie manipuliert werden können.

14.1. Strings und der ASCII-Code

Da mit dem PRINT-Befehl ja auch Daten an ein anderes Gerät gegeben werden, nämlich an den Bildschirm, versteht er die ASCII-Werte auch.

Der Tabelle im Handbuch entnehmen wir zum Beispiel den ASCII-Wert für das A, er ist 65. Um mit dieser Zahl den Buchstaben A auf den Bildschirm zu zaubern, gibt es einen weiteren Basic-Befehl:

CHR$(ASCII-Wert)

Er ist eine Abkürzung von »Character«, was auf deutsch »Zeichen« bedeutet. Das String-Symbol muß immer dahinter stehen.

Basic-Befehl Nr. 15 CHR$(X)
- wandelt die ASCII-Codezahl X in ihr entsprechendes Zeichen um
- die Zuordnung der ASCII-Codes und der Zeichen ist im Handbuch als Anhang enthalten

In Verbindung mit dem PRINT-Befehl sieht der CHR$-Befehl so aus:

```
→ PRINT CHR$(65)
```

Diese Zeile, sowohl im Direkt-Modus als auch in einer Programmzeile, bewirkt dasselbe wie PRINT"A". Sie druckt den Buchstaben A auf den Bildschirm.

Was bringt denn das, werden Sie jetzt fragen. Die gute alte Anführungszeichen-Methode ist doch viel praktischer und kürzer. Aber die Methode der ASCII-Codewerte kann dennoch recht vielseitig sein.

Mit Zahlen, auch mit Codezahlen, kann man nämlich rechnen. Geben Sie bitte folgende Zeilen ein:

```
→ 10 X=64
  20 X=X+1
  30 PRINT CHR$(X);
  40 IF X=90 THEN END
  50 GOTO 20
```

In Zeile 10 geben wir der numerischen Variablen X den Wert 64, das ist um 1 weniger als die Codezahl des Buchstaben A.

In Zeile 20 wird eine Zählschleife begonnen mit der Erhöhung von X um 1.

Zeile 30 druckt das dieser Zahl entsprechende Zeichen aus. Im ersten Durchlauf ist es das A.

Zeile 50 schließt die Zählschleife durch den Rücksprung auf Zeile 20. Dadurch werden alle Zeichen vom Codewert 65 bis 90 - das ist das Alphabet von A bis Z - ausgedruckt.

Die Prüfzeile 40 wartet, bis das Z den Wert 90 erreicht und beendet dann das Programm. Nur mit Buchstaben in Anführungszeichen wäre dieses Programm recht lang geworden.

Der Befehl CHR$ wird hauptsächlich bei der Verarbeitung von Strings, also bei Erkennen, Vergleichen und Verändern von Wörtern eingesetzt. Er gehört zu einer ganzen Gruppe von Befehlen, die alle mit der String-Verarbeitung zu tun haben. Wir werden sie alle noch kennenlernen.

Der direkte Nachbar zu CHR$ ist der Befehl

ASC (String)

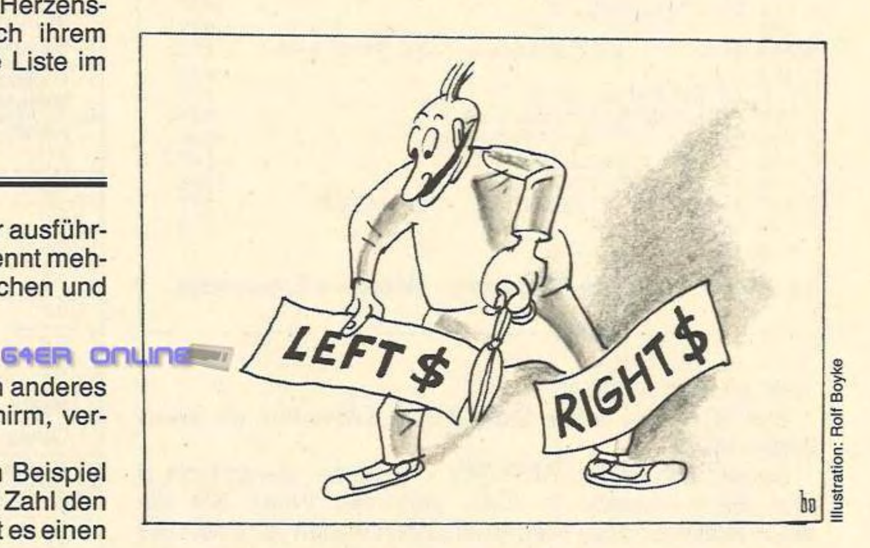


Illustration: Rolf Boyke

Er ist die exakte Umkehrung des CHR$-Befehls. ASC ist die Abkürzung von ASCII.

Erinnern Sie sich, ich habe diesen Befehl schon kurz verwandt, in Lektion 13 bei der Vorführung des ASCII-Codes.

Der Befehl wandelt das erste Zeichen des in Klammern stehenden Strings in seinen ASCII-Codewert um.

```
→ PRINT ASC("WORT")
```

Diese Zeile druckt den ASCII-Wert von W aus, es ist 87.

Basic-Befehl Nr. 16 ASC(String)
wandelt das erste Zeichen des Strings in seinen ASCII-Codewert um

Als Beispiel verwende ich die schon mehrfach geübte Frage nach Wiederholung oder Ende eines Programms, diesmal aber durch Vergleich der ASCII-Werte.

```
→ 10 PRINT"PROGRAMM-ANFANG"
   .
   .    (Programm)
   .
  100 PRINT"NOCH EINMAL (J/N) ?"
  110 GET A$:IF A$="" THEN 110
  120 X=ASC(A$)
  130 IF X=74 THEN 10
  140 IF X <> 78 THEN 100
  150 END
```

In Zeile 10 soll das Programm beginnen - welches, interessiert uns hier nicht weiter.



Am Ende dieses Programms fordert Zeile 100 zur Eingabe von JA oder NEIN auf.

Zeile 110 enthält das gewohnte Bild der GET-Schleife, die wartet, bis ein Zeichen im Tastaturpuffer erscheint.

Zeile 120 wandelt das erste Zeichen des eingegebenen Wortes in seinen ASCII-Code um. Aus der ASCII-Codetabelle wissen wir, daß dem erwarteten J die Zahl 74, dem N aber die Zahl 78 entspricht. Alle anderen Werte sollen zurückgewiesen werden.

Ist X=74, dann springt Zeile 130 auf den Anfang des Programms zurück.

Ist X nicht 78, wird durch Zeile 140 die Eingabe wiederholt.

Im Fall, daß X=78, fällt die Prüfung in Zeile 140 durch, und das Programm macht in Zeile 150 Schluß.

Auch dieses Programm ist länger als die alte Methode, aber es zeigt wenigstens die Arbeitsweise von ASC.

14.2. Das Zerteilen von Strings

Der Befehl ASC hat den Nachteil, daß er immer nur das erste Zeichen des Strings zur Umwandlung nimmt. Um ein ganzes Wort in seine einzelnen ASCII-Codewerte zu zerlegen, brauchen wir eine Methode, welche uns gestattet, einzelne Zeichen des Wortes abzuschneiden bzw. herauszupicken. Ich müßte eigentlich String statt Wort sagen, denn das alles gilt natürlich auch für Zeichen und Symbole.

Es wäre schön, wenn man dieses Abschneiden am Anfang, in der Mitte oder am Ende dieses Wortes bzw. Strings machen könnte.

Basic kann alle drei:
- LEFT$ schneidet vom linken Rand des Strings Zeichen heraus
- RIGHT$ tut dasselbe auf der rechten Seite
- MID$ pickt Teile aus der Mitte des Strings heraus

Natürlich bieten diese Befehle auch die Möglichkeit, anzugeben, wieviele Zeichen abgetrennt werden sollen.

Bei LEFT$ und RIGHT$ ist das nur eine einzige Zahlenangabe.

Bei MID$ brauchen wir zwei Zahlen. Die eine gibt an, ab wo herausgepickt werden soll, die zweite sagt wieviel.

Im Beispiel wird das schnell klar:

```
→ 10 A$= "MOTORHAUBENVERSCHLUSS"
  20 B$ = LEFT$(A$,5)
  30 PRINT B$
  40 C$ = RIGHT$(A$,7)
  50 PRINT C$
  60 D$ = MID$(A$,6,5)
  70 PRINT D$
```

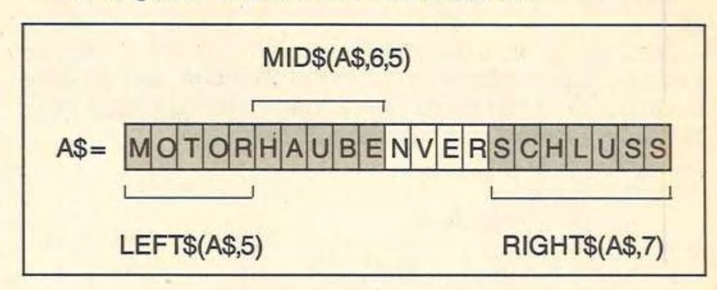
Zeile 10 legt uns die Basis mit der Zuweisung des langen Wortes an die String-Variable A$. Jetzt geht's ans Beschneiden. Mit Zeile 20 fangen wir an der linken Seite des Wortes an. Hinter dem Befehl LEFT$ steht in der Klammer zuerst der String, der zerpflückt werden soll, nämlich A$. Die zweite Angabe - nach dem Komma - gibt an, wieviele Zeichen abgeschnitten werden sollen, im Beispiel sind es 5. Diese 5 ergeben gerade den neuen String »MOTOR«, der in Zeile 30 ausgedruckt wird.

Die Schreibweise und Funktion von RIGHT$ ist identisch, halt nur auf der rechten Seite des Wortes A$ wirkend. In Zeile 40 erhält dieser Teilstring den Variablennamen C$. Zeile 50 druckt also die rechten 7 Zeichen des Strings A$ aus, was das Wort »SCHLUSS« ergibt.

Interessant wird es in Zeile 60 beim Befehl MID$. Nach der Angabe des betroffenen Strings A$ steht zuerst die Nummer des Zeichens, ab dem von links gezählt herausgeschnippelt werden soll, die zweite Zahl daneben gibt an, wieviele Zeichen es sein sollen.

In unserem Beispiel ist das sechste Zeichen von links das »H«, ab da 5 Zeichen weiter - inklusive des »H« - ergeben das Wort »HAUBE«, das in Zeile 70 ausgedruckt wird.

Das folgende Bild soll das verdeutlichen:



Basic-Befehl Nr. 17 LEFT$(X$,A)
- schneidet vom String X$ von links her A-Zeichen ab und bildet daraus einen neuen String

Nr. 18 RIGHT$(X$,A)
- macht genau dasselbe wie Nr. 17, aber von rechts her

Nr. 19 MID$(X$,B,A)
- schneidet vom String X$ von links her ab dem B-Zeichen insgesamt A-Zeichen heraus
- bei MID$ kann die zweite Zahl »A« weggelassen werden. Dann schneidet es ab dem B-Zeichen den Rest des Strings ab

Ich empfehle Ihnen, ein bißchen zu experimentieren, am besten mit nur einer Zeile

```
→ 100 PRINT LEFT$("DRACHEN",2)
  RUN 100
```

Erstaunt Sie die Schreibweise? Das sollte jetzt eigentlich nicht mehr passieren.

Der einzige Unterschied zu vorher ist, daß wir vorher den String zuerst einer String-Variablen zugeordnet haben, wodurch er im Speicher aufbewahrt wird. Hier schreiben wir den String direkt an die Stelle der Variablen, der Computer merkt ihn sich halt nicht. Aber mit RUN 100 können Sie leicht den Befehl immer wieder ausführen, nachdem Sie mit dem Cursor auf die Zahl gefahren sind und diese verändert haben.

Genauso geht es mit dem Befehl RIGHT$, weswegen ich ihn hier auslasse.

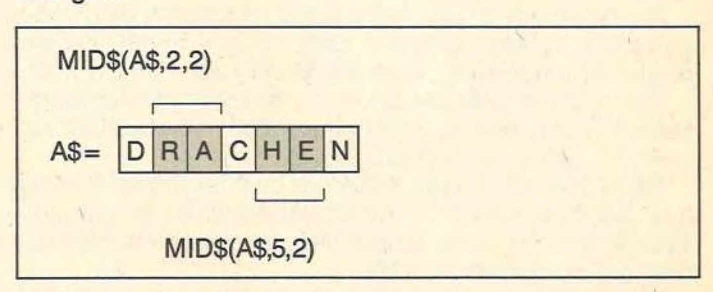
MID$ ist der kniffligste der drei Befehle.

```
→ 200 PRINT MID$("DRACHEN",2,5)
```

ergibt »RACHE«. Wenn Sie die zweite Zahl ganz weglassen, erhalten Sie »RACHEN«.

Die Frage stellt sich, ob wir aus dem Wort »DRACHEN« mit diesem Befehl zwei getrennte Teile herausholen und so zusammensetzen können, daß zum Beispiel das Wort »RAHE« entsteht.

Das geht!!



Wir müssen bloß zuerst den Teil »RA« und in einem zweiten Schritt den Teil »HE« herauszwicken und dann beide Teilstrings zu einem neuen String zusammenfügen. Die folgenden Zeilen schaffen das:

```
→ 10 A$="DRACHEN"
  20 X$=MID$(A$,2,2)
  30 Y$=MID$(A$,5,2)
```

```
40 Z$=X$+Y$
50 PRINT Z$
```

Ich glaube, das brauche ich nicht zeilenweise zu erläutern.

Jetzt soll aber noch ein bißchen Mathematik dazukommen. Es ist ja nicht zwingend vorgeschrieben, daß die Zahlen in der Klammer hinter dem String-Befehl konstant sind. Schauen Sie sich das folgende Beispiel an:

```
→ 10 A$="DRACHEN"
  20 X=X+1
  30 B$=LEFT$(A$,X)
  40 PRINT B$
  50 IF X=7 THEN END
  60 GOTO 20
```

Dieses Programm druckt uns das folgende Muster aus:

D
DR
DRA
DRAC
DRACH
DRACHE
DRACHEN

Das Geheimnis liegt in Zeile 30, in der die Zahl der abzuschneidenden Zeichen nicht konstant, sondern durch die Zählschleife mit X von 1 bis 7 hochgezählt wird.

Und jetzt kommen wir unserem früheren Wunsch, alle Buchstaben eines Wortes in ihren ASCII-Code umzuwandeln, schon näher.

Erinnern Sie sich, diese Umwandlung liefert uns der ASC-Befehl, aber immer nur für den ersten Buchstaben eines Wortes.

Was wir brauchen, ist so ein Muster:

DRACHEN
RACHEN
ACHEN
CHEN
HEN
EN
N

Mit LEFT$ geht das nicht, denn wir schneiden ja an der rechten Seite ab. Außerdem fangen wir mit der vollen Wortlänge von 7 an und reduzieren sie laufend.

Also, der RIGHT$-Befehl muß her, kombiniert mit einer Zählschleife, die aber rückwärts zählt:

```
→ 10 A$="DRACHEN"
  20 X=7
  30 B$=RIGHT$(A$,X)
  40 PRINT B$
  50 X=X-1
  60 IF X=0 THEN END
  70 GOTO 30
```

Die rückwärts zählende Schleife wird durch die Festlegung des Anfangswertes in Zeile 20 und durch das laufende Vermindern der Zählvariablen in Zeile 50 gebildet.

Um noch den ASCII-Codewert der jeweils ersten Buchstaben zu erhalten, müssen wir nur Zeile 40 erweitern zu:

```
→ 40 PRINT B$ "=" ASC(B$)
```

Nach dem bisherigen Ausdruck des Teilstrings B$ folgt das Gleichheitszeichen – als String zwischen Anführungszeichen gesetzt – und danach der umgewandelte Wert des ersten Buchstabens von B$.

Ich muß Ihnen gestehen, daß das alles nur zur Übung war. Denn am elegantesten geht es mit dem Befehl MID$.

Wir kehren wieder zur hochzählenden Schleife zurück und zwicken von links her der Reihe nach die Buchstaben heraus, immer nur einen, mit dem Befehl MID$(A$,X,1)

```
→ 10 A$="DRACHEN"
  20 X=X+1
  30 B$=MID$(A$,X,1)
  40 PRINT B$ "=" ASC(B$)
  50 IF X=7 THEN END
  60 GOTO 20
```

Es gibt noch einen STRING-Befehl, der uns das Zählen der Zeichenzahl für die Prüfung in Zeile 50 abnimmt. Er heißt

LEN(String)

Er ist eine Abkürzung von Length, was auf deutsch »Länge« heißt. Er wird seinem Namen gerecht und bestimmt die Länge des Strings.

Wenn Sie direkt eingeben:

```
→ PRINT LEN("DRACHEN")
```

dann erhalten Sie die Zahl 7 als Resultat.

Um ihn im obigen letzten Programmbeispiel einzusetzen, brauchen wir lediglich die Zeile 50 ändern:

```
→ 50 IF X=LEN(A$) THEN END
```

So prüft sie X so lange, bis es den Wert von LEN(A$), nämlich 7 erreicht hat.

Basic-Befehl Nr. 20 LEN(String)
- gibt die gesamte Länge des Strings einschließlich der Leerzeichen und Steuerzeichen als Zahl aus
- in der Klammer kann ein String oder aber eine String-Variable sein, der ein String zugewiesen worden ist

14.3. Die Addition von Strings

In Abschnitt 6.2, bei der Diskussion der String-Variablen, haben wir bereits mit zusammengehängten Strings experimentiert, unter Verwendung des PRINT-Befehls:

```
110 A$="HOLZ"
115 B$="FEUER"
120 PRINT A$B$
```

Das Resultat der Zeile 120 ist das Wort HOLZFEUER. Dabei ist es egal, ob Sie in Zeile 120 zwischen den beiden Stringvariablen ein Semikolon, einen Zwischenraum oder, wie oben, beide aneinander schreiben.

Wenn wir aber dem »HOLZFEUER« eine eigene Variable zuordnen wollen, müssen wir A$ und B$ addieren, unter Verwendung des »+«-Zeichens.

```
125 C$=A$+B$
130 PRINT C$
```

Zeile 120 und Zeile 130 ergeben das gleiche Resultat, nur hat sich der Computer die neue Stringvariable C$ gemerkt; das Resultat der Addition ist dadurch auch noch später verfügbar.

Nicht ganz so selbstverständlich wie Strings mit Buchstaben sind Strings, die aus Zahlen bestehen.

```
135 X$="7"
140 Y$="5"
145 Z$=X$+Y$
150 PRINT Z$
```

Zeile 150 druckt die Zahl 75 aus. Bemerkenswert ist dabei, daß in der String-Form die Ziffern ohne den sonst üblichen freien Platz für ein eventuelles negatives Vorzeichen gedruckt werden.

Wir haben in Lektion 13 gelernt, Steuerzeichen wie zum Beispiel »CURSOR-LINKS«, »REVERSE-ON« innerhalb von Anführungszeichen in Ihre Programme einzubauen. Nun, da diese auch Strings darstellen, können wir sie ebenfalls mit anderen Strings zusammensetzen.

```
155 L$=" {CTRL+RVS-ON}"
160 M$=" {CTRL+RVS-OFF}"
165 PRINT L$+B$+M$+A$
```

Zeile 165 druckt das Wort FEUERHOLZ aus, wobei der Wortteil FEUER in reverser Darstellung erscheint. Zu guter Letzt sollen die Zeilen 170 bis 195 zeigen, daß auch die Grafikzeichen, die vorn auf den Tasten stehen und auf die mit der SHIFT- oder der CBM-Taste umgeschaltet wird, als Strings addiert werden können.



```
170 Q$=" {SHIFT+O/CBM+Y/SHIFT+P} "
175 R$=" {3mal CRSR-LINKS} "
180 S$=" {CRSR-DOWN} "
185 T$=" {SHIFT+L/CBM+P/SHIFT+@} "
190 U$=Q$+R$+S$+T$
195 PRINT U$
```

In Zeile 190 wird eine Stringvariable U$ definiert, die aus 6 Zeichen und mehreren Cursorbewegungen besteht, wodurch ein Rechteck gezeichnet wird. In Zeile 195 wird dieses Rechteck ausgedruckt.

Wenn wir die Zeile 195 so abändern:

```
195 PRINT U$;:GOTO 195
```

wird in einer »ewigen« Schleife das Rechteck diagonal über den Bildschirm gedruckt.

Die aus einzelnen Zeichen zusammengesetzte Form wird wie ein einzelnes neues Zeichen behandelt!

Diese Additionsmethode für Strings kann in vielen Programmarten sehr sinnvoll eingesetzt werden.

FAZIT
1. Durch Addition von einzelnen Strings werden neue, längere Strings gebildet.
2. Aus einzelnen Buchstaben pro Variable entstehen so einzelne Wörter pro Variable, aus diesen wiederum ein ganzer Satz pro Variable.
3. Eine String-Variable kann maximal 255 Zeichen enthalten.

Eine derart zusammengesetzte String-Variable kann aus Buchstaben, grafischen Zeichen, Ziffern und Steuerzeichen bestehen.

14.4. Der Größenvergleich von Strings

Beim Stringvergleich mit > und < muß der Programmierer wissen, was einen »größeren« String ausmacht. Ich möchte es Ihnen mit einem kleinen Versuch deutlich machen.

```
310 INPUT "STRING #1";A$
320 IF A$="@" THEN END
330 INPUT "STRING #2";B$
```

Wir geben also zwei Strings in den Zeilen 310 und 330 ein. Zeile 320 erlaubt uns den Aussprung aus der Schleife mit der Klammeraffen-Taste.

Jetzt vergleichen wir die beiden Strings:

```
340 IF A$ > B$ THEN PRINT "#1"
350 IF A$ < B$ THEN PRINT "#2"
360 GOTO 310
```

Die Vergleiche in Zeile 340 und 350 wählen beide den größeren String zum Ausdrucken aus.

Lassen Sie das Programm laufen und experimentieren Sie ein bißchen.

Sie werden zum Beispiel sehen:
»A« ist größer als »B«
»ACB« ist größer als »ABC«
»TISCHE« ist größer als »TISCH«
»WORT 2« ist größer als »WORT 1«
Ich will Ihnen verraten, was der Computer macht.

Er vergleicht die ASCII-Codewerte der einzelnen Buchstaben von links aus. Zur Erinnerung, der ASCII-Code ist eine Zahl, mit der der Computer ein Zeichen intern kennzeichnet (siehe Lektion 13).

Das A hat den ASCII-Wert 65, B den Wert 66, C den Wert 67. Sowohl bei ACB als auch bei ABC ist das erste Zeichen gleich, aber von den zweiten Zeichen ist C größer als B. Im Beispiel TISCHE-TISCH macht das zusätliche E den Unterschied im String 2.

Beim Vergleich von WORT 2 mit WORT 1 hat die Ziffer 2 einen höheren ASCII-Codewert als die Ziffer 1.

Das Beispiel oben zeigt, daß natürlich auch Zahlen-Strings größer oder kleiner sein können, genauso wie gemischte Strings (WORT 2).

FAZIT
1. Werden Strings miteinander verglichen, vergleicht der Computer schrittweise die ASCII-Codewerte der einzelnen Zeichen. Die erste Ungleichheit entscheidet, wobei dann der String mit dem höheren ASCII-Codewert »größer« ist.
2. Da der Stringvergleich sich auf ASCII-Codewerte bezieht, wird kein Unterschied gemacht zwischen Buchstaben, Zeichen oder Ziffern.

14.5. Verwandlung von Strings und Zahlen

Im Abschnitt 14.3 haben wir gesehen, daß Zahlen als Strings verarbeitet werden, wenn sie in Anführungszeichen stehen. Das hat unter anderem den Vorteil, daß kein Leerzeichen für das Vorzeichen reserviert wird.

```
10 A$="123"
15 PRINT A$
```

druckt die Zahl ganz an den linken Rand.

Es gibt zwei Befehle, die uns erlauben, Strings in Zahlen und Zahlen in Strings zu verwandeln:

Der eine der beiden heißt VAL(A$). Sein Name ist von VALUE (Wert) abgeleitet. Er wandelt A$ in einen Zahlenwert um, falls überhaupt in A$ eine Zahl vorkommt.

```
20 PRINT VAL(A$)
```

Diese Zeile druckt ebenfalls die Zahl 123 aus, aber eben als Zahl, das heißt mit einer Leerstelle vor ihr.

Ist in dem String keine Zahl enthalten, dann wird der Wert Null ausgegeben.


```
25 B$="ABC"
30 PRINT VAL(B$)
```

Wenn wir statt »ABC« den String »A2C« schreiben, erhalten wir immer noch Null, weil der String mit einem Buchstaben anfängt.

Ist B$ aber »12C«, dann ergibt der VAL-Befehl die Zahl 12.

Basic-Befehl Nr. 21 VAL(A$)
- Der Befehl VAL(A$) liefert den numerischen Wert von A$
- Dieser String kann sowohl in dem $-Zeichen als auch in Anführungszeichen geschrieben sein
- VAL(A$) fragt den String A$ Zeichen für Zeichen nach einer Zahl ab. Ab dem ersten Zeichen, das keine Zahl ist, stoppt er die Suche und verwendet den Wert des bisherigen Teils weiter
- Wenn der String A$ keine Zahlen enthält, wird der Wert 0 gebildet

Die Umkehrung von VAL ist der Befehl STR$(X), abgeleitet von STRING. Er wandelt die Zahl X in einen String um. Wozu das gut ist, zeigen uns die nächsten Zeilen:

```
35 X=123
40 Y=456
45 PRINT X+Y
```

Das Resultat der Zeile 45 ist natürlich die Summe der beiden Zahlen, also 579, mit einer Leerstelle ausgedruckt.

```
50 PRINT X;Y
```

Diese Zeile dagegen setzt beide Zahlen nebeneinander, getrennt durch zwei (!) Leerstellen, eine für das Vorzeichen, die zweite für die Trennung zweier unabhängiger Zahlen:

```
123  456
```

Diese zweite Trennung heben wir auf mit der Verwendung des STR$-Befehls:

```
55 PRINT STR$(X);STR$(Y)
```

Wir erhalten:

```
123 456
```

Aber Sie sehen, daß X und Y immer noch Zahlen sind, obwohl sie über STR$ als Strings behandelt werden.
Erst die Zeile:

```
60 PRINT "123";"456"
```

macht echte Strings aus den Zahlen und vermeidet alle Zwischenräume.

```
123456
```

Leider können wir in dieser Version mit den Zahlen nicht mehr rechnen. Mit STR$ geht das aber, was die nächsten Zeilen 65 bis 75 zeigen.

```
65 FOR Z=0 TO 2
70 PRINT STR$(X+Z);STR$(Y)
75 NEXT Z
```

Wir erhalten die Zahlenreihen

```
123 456
124 456
125 456
```

Basic-Befehl Nr. 22 STR$(X)
- der Befehl STR$(X) bildet aus dem Wert der Zahl X einen String
- dieser String behält die Leerstelle für das Vorzeichen

LEKTION 15: Ein Programm mit den String-Befehlen

Ich möchte mit Ihnen zusammen ein zweites Programm entwickeln, welches hauptsächlich die Stringbefehle verwendet und Ihnen ein bißchen Erfahrung in der String-Verarbeitung gibt.

Aufgabe:

Im ersten Teil des Programms sollen eingegebene Wörter in einen Geheimcode umgewandelt werden. Die verwandelten Wörter stehen alle auf dem Bildschirm.

Per Tastendruck soll dann der zweite Programmteil angesprungen werden, in dem die einzelnen Codewörter wieder in ihre ursprüngliche Form zurückgewandelt (dekodiert) werden.

Der übersichtlichen schrittweisen Darstellung zuliebe verzichte ich auf ein Flußdiagramm. Wir »hacken« uns also durch.

Die Eingabe der Wörter machen wir zuerst über den GET-Befehl. Sie wissen, daß der GET-Befehl immer nur ein Zeichen annimmt, aber keine ganzen Wörter. Wir müssen also erst eine Schleife bauen, um die einzelnen Zeichen zu einem Wort zusammenzusetzen.

```
100 GET W$: IF W$="" THEN 100
120 PRINT W$;
130 A$=A$+W$
150 GOTO 100
```

Zeile 100 ist die übliche Eingabe-Warteschleife mit GET.

Zeile 120 druckt jedes einzelne Zeichen W$ mit dem Semikolon aneinandergeklebt aus.

Zeile 130 ist etwas Neues. Es wird ein String mit Namen A$ dadurch gebildet, daß zu seinem jeweiligen Zeichen – am Anfang ist keines da – das gerade eingegebene Zeichen W$ dazugenommen wird. Auf diese Weise entsteht ein ganzes Wort A$.

Der Rücksprung in Zeile 150 läßt dieses Wort unendlich lange werden.

Wir brauchen also am Ende des Wortes einen Aussprung aus der Schleife. Ich schlage vor, das Wort mit der RETURN-Taste zu beenden. Deshalb müssen wir in Zeile 140 den ASCII-Code der RETURN-Taste abfragen. Er ist 13.

```
140 IF W$=CHR$(13) THEN 170
170 PRINT"**"
```

Die Abfrage kennen wir schon. Sie springt nach 170, wo vorläufig zum Ausprobieren zwei Sterne ausgedruckt werden.

Codier-/Decodier-Algorithmus

»Algorithmus« ist die Bezeichnung für einen logischen Vorgang oder eine Formel, in unserem Fall die Vorschrift, nach der das Wort A$ verschlüsselt werden soll.

Ich habe einen sehr leichten Algorithmus gewählt:

Der ASCII-Codewert jedes Zeichens wird um die Zahl seines Platzes im Wort erhöht, der erste Buchstaben um 1, der zweite Buchstaben um 2 und so weiter.

Das Decodieren geht entsprechend, indem immer die Platz-Ziffer abgezogen wird.

Also müssen wir zuerst das Wort A$ in seine ASCII-Codewerte umwandeln. Das haben wir schon einmal gemacht.

In Zeile 160 gebe ich ein Wort für A$ vor, aber nur zum Ausprobieren.

```
160 A$="WERT"
170 X=X+1
180 A=ASC(MID$(A$,X,1))
190 B=A+X
195 PRINT B
220 IF X <> LEN(A$) THEN 170
```

A$ in Zeile 160 sei also das fertig eingegebene Wort.

Die Zeilen 170 und 220 bilden eine Zählschleife, deren Variable X von 1 bis zur Länge des eingegebenen Wortes A$ zählt und am Ende weiterspringt. In unserem Beispiel ist LEN(A$)=4 Buchstaben.

Zeile 180 kennen Sie schon. Sie wandelt jeden Buchstaben des Wortes A$ – einzeln wegen der letzten Ziffer 1 – von links ab dem X-Zeichen in seinen ASCII-Code um.

Zeile 190 ist die eigentliche Verschlüsselung oder Codierung. Hier wird eine Variable B dadurch gebildet, daß zum ASCII-Wert »A« des gerade umgewandelten Zeichens »A$« der jeweilige Wert von X dazugezählt wird.

Zeile 195 ist eingeschoben, um die neuen ASCII-Werte auszudrucken.

Starten Sie diesen Teil mit RUN 160 und Sie erhalten die neue Zahlenfolge 88, 71, 85 und 88.

Jetzt nehmen wir Zeile 160 heraus, oder beser noch, wir wandeln sie in eine REM-Zeile zur Schönschrift um.

Sie können jetzt mit RUN das Ganze laufen lassen, nämlich Eingabe eines Wortes bis zum Ausdruck der neuen Codezahlen. Jetzt allerdings erhalten Sie immer eine Zahl mehr, als Zeichen eingegeben wurden. Diese zusätzliche Zahl entsteht durch die RETURN-Taste.

Die nächsten drei Zeilen wandeln die verschlüsselten Codezahlen »B« in Buchstaben »B$« zurück und setzen sie zu einem verschlüsselten Wort »C$« zusammen.

```
→ 200 B$=CHR$(B)
  210 C$=C$+B$
  240 PRINT C$
```

Mit RUN erhalten Sie jetzt auch das verschlüsselte Wort ausgedruckt.

Um mehrere Wörter eingeben zu können, bilden wir mit dem GOTO-Befehl der Zeile 260 eine Schleife. Zeile 195 kann jetzt entfernt werden.

```
→ 195
  260 GOTO 100
```

Ein neuer Versuch klappt allerdings nicht.

Die Eingabe eines zweiten Wortes stößt auf Schwierigkeiten. Der Grund dafür liegt darin, daß das erste Wort A$ noch vorhanden ist. Wir müssen es einfach löschen. Und weil wir gerade beim Korrigieren sind: Auch die Zählvaria-

ble X muß vor der nächsten Eingabe auf 0 gesetzt werden. Der letzte Fehler besteht darin, daß in Zeile 220 der LEN-Befehl immer ein Zeichen mehr mißt, weil ja die RETURN-Taste am Ende des eingegebenen Wortes für ihn auch als Zeichen gewertet wird.

Hier bietet sich zusätzlich die Gelegenheit, für die Schönschrift der Ausgabe etwas zu tun. Mir gefällt nämlich nicht, daß beim Ausdrucken zwischen den codierten Wörten kein Zwischenraum gelassen wird. Ich schiebe daher die Zeile 230 vor den PRINT-Befehl, um an den bestehenden String »C$« - das codierte Wort - ein Leerzeichen mit dem ASCII-Code 32 anzuhängen.

Die folgenden Zeilen korrigieren das alles.

```
→ 220 IF X < >  LEN(A$)-1 THEN 170
  230 C$=C$+CHR$(32)
  250 A$=""
  260 X=0: GOTO 100
```

In Zeile 220 prüft der IF-THEN-Befehl auf ein Zeichen weniger, als das Wort plus RETURN-Taste lang ist. Zeile 250 zeigt, wie man eine Stringvariable durch Zuweisung eines Nullstrings löscht. Dasselbe tut Zeile 260 mit der numerischen Variablen X.

Jetzt läuft der erste Programmteil, der aus einem eingegebenen Wort ein verschlüsseltes Wort baut. Wir erhalten zum Beispiel für das Wort »WERT« den Code »XGUX«.

Das Decodieren stelle ich mir so vor, daß der Benutzer per Tastatur die Codes eingibt, welche er entschlüsseln will. Das wollen wir zur Übung diesmal mit INPUT machen.

```
→ 280 INPUT E$
```

Die nächsten Zeilen gleichen weitgehend den Zeilen 170 bis 260, da der Vorgang ja fast identisch ist. Eine Ausnahme ist die Zeile 190, da wir zum Decodieren ja die Zählvariable X abziehen müssen. Außerdem habe ich die beiden Zeilen 200 und 210 zu einem Ausdruck zusammengefaßt. Und schließlich braucht dem LEN-Befehl keine 1 abgezogen werden, da beim INPUT-Befehl im Gegensatz zum GET die eingegebene RETURN-Taste nicht zum Wort dazugerechnet wird. Also sieht das Decodieren so aus:

```
→ 280 INPUT E$
  290 X=X+1
  300 E=ASC(MID$(E$,X,1))
  310 D=E-X
  320 D$=D$+CHR$(D)
  330 IF X < > LEN(E$) THEN 290
  340 D$=D$+CHR$(32)
  350 PRINT D$
  360 E$=""
  370 X=0:GOTO 280
```

Mit RUN 280 können Sie diesen Programmteil separat testen. Jetzt müssen wir die beiden Teile noch zusammenbinden. Ich stelle mir vor, daß es praktisch ist, nach einigen eingegebenen Wörtern diese zu decodieren. Das Signal dazu soll die Funktionstaste <F1> sein. Die Arbeitsweise der Funktionstasten haben wir noch nicht durchgenommen. Springen Sie bitte auf die Lektion 16, bevor Sie mit diesem Programm weitermachen.

Wenn die Funktionstaste gedrückt wird, soll das Programm vom Kodierteil auf den Decodierteil springen.

Diese Prüfung müssen wir ganz am Anfang des Codierteils vornehmen, und die GET-Eingabeschleife bietet sich dazu in hervorragender Weise an. Wir schieben also die folgenden drei Zeilen ein:

```
→ 110 IF W$=CHR$(133) THEN 270
  270 PRINT"DEKODIEREN"
  285 IF E$="ENDE" THEN END
```

Zeile 110 verwendet den ASCII-Code von 133 der F1-Taste, um auf die Zeile vor dem INPUT-Befehl zu springen (270), wo ein Hinweis ausgedruckt wird.

Die Zeile 285 beendet auf simple Weise das Programm. Das einzige, was nicht passieren darf ist, daß das Wort »ENDE« als codierter Wert auftritt.

Zur Bedienung des Programms ist zu sagen, daß Sie beim Decodieren lediglich die auf dem Bildschirm stehenden verschlüsselten Wörter einzeln ablesen und eintippen müssen.

Wem das zu mühsam ist, soll ein Programm für die automatische Decodierung schreiben !

FAZIT

1. Eine String-Variable kann dadurch »gelöscht« werden, daß man ihr einen Null-String zuweist mit dem Befehl A$= "".
   Das entspricht dem Nullsetzen einer numerischen Variablen mit X=0
2. Funktionstasten und andere Steuertasten, die nicht auf dem Bildschirm erscheinen, können nur mit dem GET-Befehl abgefragt werden.
3. Ein String kann durch Hinzufügen von anderen Strings zu einem größeren String gleichen Namens erweitert werden: A$=A$+B$ 3.

In Listing 2 ist das komplette Programm wiedergegeben, auch hier mit den Prüfsummen der Eingabehhilfe »Checksummer 64 V3«.

Und jetzt kommen im Nachtrag die Funktionstasten an die Reihe.

LEKTION 16: Die Funktionstasten

Ich möchte gern einen Rücksprung machen zu der Tabelle der Steuertaste und ihrer ASCII-Codewerte. Da sehen Sie als untersten Block die Funktionstasten <F1> bis <F8> angegeben. In diesem Kurs werden sie da praktisch zum ersten Mal erwähnt. Jetzt wird es langsam Zeit, näher darauf einzugehen.

```
10 PRINT"**** KODIEREN/DEKODIEREN ****      <206>
20 :                                        <252>
30 :                                        <006>
100 GET W$:IF W$="" THEN 100                <160>
110 IF W$=CHR$(133) THEN 270                <068>
120 PRINT W$;                               <065>
130 A$=A$+W$                                <248>
140 IF W$=CHR$(13) THEN 170                 <142>
150 GOTO 100                                <078>
160 REM----------------------------         <253>
170 X=X+1                                   <198>
180 A=ASC(MID$(A$,X,1))                     <070>
190 B=A+X                                   <219>
200 B$=CHR$(B)                              <105>
210 C$=C$+B$                                <123>
220 IF X<>LEN(A$)-1 THEN 170                <233>
230 C$=C$+CHR$(32)                          <011>
240 PRINT C$                                <178>
250 A$=""                                   <023>
260 X=0:GOTO 100                            <208>
265 REM-----------------------------        <254>
270 PRINT"DEKODIEREN"                       <014>
280 INPUT E$                                <156>
285 IF E$="ENDE" THEN END                   <191>
290 X=X+1                                   <064>
300 E=ASC(MID$(E$,X,1))                     <212>
310 D=E-X                                   <189>
320 D$=D$+CHR$(D)                           <246>
330 IF X<>LEN(E$) THEN 290                  <117>
340 D$=D$+CHR$(32)                          <159>
350 PRINT D$                                <042>
360 E$=""                                   <151>
370 X=0:GOTO 280                            <199>
© 64'er
```

Listing 2. Kodieren und Dekodieren mit den String-Befehlen

Wenn Sie eine der Funktionstasten drücken, passiert bekanntlich gar nichts - oder doch?

Nun, sie haben einen eigenen ASCII-Code. Mit einem kleinen Programm, das wir bei der Besprechung der ASCII-Codes verwendet haben, können wir diese Behauptung hier noch einmal überprüfen. Das Programm sah so aus:

```
10 INPUT A$
20 A=ASC(A$)
30 PRINT A
40 GOTO 10
```

Nicht nur die Funktionstasten, sondern auch alle anderen Steuertasten geben so ihren ASCII-Code preis.

Diesen Code können wir zur Abfrage der Tasten, ob sie nämlich gedrückt worden sind, verwenden.

Fügen Sie bitte die folgende Zeile 35 ein:

```
35 IF A=65 THEN PRINT "SERVUS"
```

Das Programm funktioniert wie vorher, nur wenn wir das »A« drücken, dessen ASCII-Code 65 ist, wird der freundliche Gruß ausgedruckt. Statt 65 können Sie jeden beliebigen Codewert verwenden und damit das Drücken der entsprechenden Taste abfragen.

Halt! Das stimmt nicht ganz. Denn wenn Sie in Zeile 35 den Codewert einer der Funktionstasten eingeben, rührt sich kein SERVUS.

Warum nicht?

Vielleicht erinnern Sie sich, daß im Unterschied zum GET-Befehl, der im Tastaturpuffer nachschaut, ob dort ein Zeichen abgespeichert ist, der INPUT-Befehl alle eingegebenen Zeichen erst auf den Bildschirm bringt und sie von dort dann weiterverarbeitet.

Das ist des Rätsels Lösung, denn die Funktionstasten hinterlassen halt keine Spur auf dem Bildschirm, genauso wenig wie alle anderen Steuertasten. Das heißt, zu deren Abfrage müssen wir den GET-Befehl nehmen.

Ändern wir also die Zeile 10 ab:

```
10 GET A$:IF A$="" THEN 10
```

Jetzt geht es!!

Zur Demonstration der Abfrage der Funktionstasten habe ich unser kleines Programm wie folgt erweitert:

```
10 GET A$:IF A$="" THEN 10
20 A=ASC(A$)
30 PRINT A
35 IF A=133 THEN PRINT CHR$(158)
36 IF A=134 THEN PRINT CHR$(154)
37 IF A=135 THEN PRINT CHR$(19)
38 IF A=136 THEN PRINT CHR$(147)
40 GOTO 10
```

Die Zeilen 35 bis 38 bilden einen Abfrageblock für die vier Funktionstasten <F1>, <F3>, <F5> und <F7>.

Wenn ihr ASCII-Code auftritt, dann drucken sie das Zeichen des hinter dem jeweiligen CHR$-Befehl stehenden Codewertes aus. Da diese Codewerte aber die Werte der Steuerzeichen - der Reihe nach - GELB, HELLBLAU, CURSOR HOME und BILDSCHIRM LÖSCHEN sind, werden diese Funktionen ausgeführt.

Das ist also das Kochrezept zur Nutzbarmachung der Funktionstasten. Aber auch alle anderen Tasten können so abgefragt und zur Steuerung irgendwelcher Programmschritte verwendet werden. Wir werden diese Methode auch noch benützen.

Übrigens, wenn Sie nur eine einzige Taste abfragen, deren Codewert aber nicht ausdrucken beziehungsweise nicht weiterverwenden, geht die Abfrage von oben dadurch schneller, daß Sie den String gar nicht erst lange umwandeln:

```
10 GET A$: IF A$="" THEN 10
35 IF A$=CHR$(133) THEN PRINT CHR$(150)
```

Die Prüfzeile 35 vergleicht einfach den eingegebenen String A$ mit dem String, der dem ASCII-Code 133 entspricht - der CHR$-Befehl macht das möglich. Nur bei vielen Abfrage-IFs ist das Tippen der vielen CHR$ mühsam und die ASC-Methode von vorher hat ihre Berechtigung.

LEKTION 17: Daten aus dem Keller holen

Es gibt Situationen im Programmierleben, in denen es wünschenswert wäre, auf einen ganzen Stall voll Zahlen oder Wörter zurückgreifen zu können, ohne sie jedesmal per INPUT oder GET mühsam in den Computer eingeben zu müssen. Einmal eingegeben, sollten sie immer zur Verfügung stehen.

Diese Anforderung zeigt deutlich in Richtung »Speicherung von Daten« im Computer. Nun, wenn wir einer String-Variablen in einer Befehlszeile einen String zuordnen, wird dieser bekanntlich auch gespeichert. Aber das würde dazu führen, daß wir beispielsweise 5 Zuordnungen machen müßten, um 5 Strings zu speichern, etwa so:

```
10 A$="BUCH"
20 B$="SCHULTER"
30 C$="SCHIFF"
40 D$="TISCH"
50 E$="MALEREI"
```

Bei 5 Strings geht es ja noch, aber bei 30 oder gar bei 200? Diesen Notstand stellt BASIC ab mit dem Befehl

DATA

Hinter diesem Befehl können in einer Befehlszeile so viel Zahlenwerte oder Strings geschrieben und dadurch gespeichert werden, wie in einer Programmzeile Platz haben.

Alle Eintragungen in einer DATA-Zeile müssen durch Kommata getrennt sein.

Das Beispiel oben, das Sie bitte vor der Eingabe der nächsten Zeilen mit NEW löschen sollten, sieht jetzt so aus:

```
10 DATA BUCH,SCHULTER,SCHIFF,TISCH,MALEREI
```

Sie sehen, die Strings brauchen nicht in Anführungszeichen stehen, mit einer Ausnahme. Diese ist weiter unten im Kasten beschrieben.

Eine DATA-Zeile mit Zahlen sieht so aus:

```
20 DATA 25,123,225,16,24
```

Jetzt muß noch geklärt werden, wie man eigentlich die gespeicherten Daten herausholen kann.

Nun, das geht erstens immer der Reihe nach, und zweitens mit dem Basic-Befehl

READ

was leicht und treffend mit »LESEN« übersetzbar ist. Probieren wir es:

```
40 READ A$
50 PRINT A$
```

Nach RUN druckt das Programm das erste der Wörter in der DATA-Zeile, nämlich »BUCH« aus. An die anderen kommen wir nicht heran.

Wenn Sie in Zeile 40 und 50 die Variable in eine numerische Variable verwandeln, kommen wir trotzdem nicht an die Zahlen. Zusätzlich bestraft uns der Computer mit einer Fehlermeldung.

Bauen wir also eine Schleife ein, um den READ-Befehl fünfmal zu wiederholen.

```
30 FOR X=1 TO 5
60 NEXT
```

Jetzt tauchen alle 5 Wörter auf. Eine Erhöhung der Schleifenvariablen X auf 6 quittiert der Computer nach RUN wieder mit einer Fehlermeldung, da wir mit der Stringvariablen A$ in den Bereich der Zahlen eingedrungen sind, wodurch Variablentyp und Wert nicht zusammenpassen.

Eine zweite Schleife schafft Abhilfe:

```
→ 70 FOR X=1 TO 5
  80 READ A
  90 PRINT A
  100 NEXT
```

Wenigstens soweit haben wir es gebracht, daß wir alle gespeicherten Daten aus dem Speicher holen können.

Erproben Sie noch durch eine Erhöhung des Endwertes der Schleife in Zeile 70 auf 10, was passiert, wenn wir mehr Daten READen wollen, als vorhanden sind. Sie werden merken, daß das nicht geht, weniger dagegen schon.

Wir wollen uns jetzt auf nur einen Datentyp beschränken. Löschen Sie bitte die Zeilen 70 bis 100.

Ein weiteres Experiment besteht darin, den READ-Vorgang endlos zu wiederholen mit einer neuen Zeile 70.

```
→ 10 DATA BUCH,SCHULTER,SCHIFF,TISCH,MALEREI
  30 FOR X=1 TO 5
  40 READ A$
  50 PRINT A$
  60 NEXT
  70 GOTO 30
```

Auch dieser Versuch schlägt fehl.

Der Grund dafür liegt im Verfahren des READ-Befehls, nämlich einen internen Zähler mitlaufen zu lassen, der anzeigt, auf das wievielte Wort der READ-Befehl zugreifen muß. Und im obigen Fall ist dieser Zähler durch die Wiederholung mit GOTO 30 über die vorhandenen 5 Wörter hinausgelaufen.

Ein zu READ-DATA gehörender Basic-Befehl stellt diesen Zähler wieder an seinen Anfang zurück. Der Befehl lautet:

RESTORE

was soviel heißt wie »wiederherstellen«. Oben vor den GOTO-30-Befehl gestellt, setzt RESTORE in der Tat alles zurecht:

```
→ 65 RESTORE
```

Die jetzt erreichte Endlos-Schleife kann mit der STOP-Taste abgewürgt werden.

Basic-Befehle Nr. 23, 24 und 25 READ, DATA und RESTORE

- mit DATA können beliebig viele Zahlen und Strings in einem Programm gespeichert werden
- sie stehen hinter dem DATA-Befehl, durch Kommata voneinander getrennt
- Strings brauchen nicht zwischen Gänsefüssen stehen, es sei denn, sie enthalten als Teil des Strings ein Komma oder ein Semikolon. Da diese besondere Steuerzwecke erfüllen, muß in diesem Fall der String zwischen Gänsefüsse gesetzt werden
- mit dem Befehl READ werden die Daten der Reihe nach gelesen. Datentyp und Variablentyp müssen einander entsprechen
- sollen die Daten mehrfach ausgelesen werden, muß mit RESTORE der Anfangszustand des Auslesens hergestellt werden

Ein lustiges Experiment will ich Ihnen noch zeigen, bevor wir ein weiteres Programm angehen.

Schreiben Sie die FOR-NEXT-Zeile 30 so wie angegeben und vertauschen Sie NEXT mit PRINT A$.

```
→ 30 FOR X=1 TO INT(RND(0)*5+1
  40 READ A$
  50 NEXT
  60 PRINT A$
  70 RESTORE
  80 GOTO 30
```

Zeile 30 müßte Ihnen bekannt vorkommen. Die obere Grenze der Schleife wird jetzt per Zufall bestimmt. Sie kann nach unserem Kochrezept von früher die Werte 1 bis 5 annehmen, aber wie gesagt, vom Zufall bestimmt.

Da PRINT nach der Schleife kommt, druckt er nur das zuletzt gelesene Wort aus. Wir treffen somit eine zufällige Auswahl aus der Menge der gespeicherten Wörter, was wir beim nächsten Programm verwenden werden.

LEKTION 18: Ein Programm mit READ-DATA

Das Programm stammt ebenfalls aus dem schon mehrfach zitierten Buch »Computerspiele und Knobeleien« von Rüdeger Baumann.

Aufgabe:

- Der Computer hat viele Begriffe gespeichert. In unserem Fall sind es 14 Sportarten.
- Per Zufallsgenerator wird einer der Begriffe ausgewählt. Nur seine Länge wird dem Spieler durch Punkte bekanntgegeben.
- Der Spieler rät einen Buchstaben.
- Das Programm vergleicht diesen Buchstaben mit allen Buchstaben des Begriffes und schreibt den Buchstaben auf die richtigen Plätze, falls es eine Übereinstimmung feststellt.
- Der Vorgang wird so lange wiederholt, bis das Wort erraten ist.
- Zum Abschluß gibt der Spieler das komplette Wort ein.

Ich möchte das Programm blockweise mit Ihnen durchgehen.

Zeile 10 löscht den Bildschirm – diesmal mit dem ASCII-Code!

Die Zeilen 80 bis 180 enthalten mit PRINT-Befehlen die Spielanleitung.

In den DATA-Zeilen 730 bis 760 sind die Sportarten gespeichert.

In den Zeilen 240 bis 260 wird in der vorher schon verwendeten Art und Weise per Zufall eine Sportart U$ ausgewählt.

Zeile 290 mißt die Länge des Wortes U$. Die Zeile 300 erfindet ein Hilfswort H$, welches am Anfang nur aus Punkten besteht, und zwar aus genau so vielen, wie der Sportbegriff lang ist.

Jetzt folgt der erste Versuch.

Zeile 370 druckt als Eingabebereitschaftszeichen (»Prompt« genannt) das punktierte Hilfswort H$ aus, um die Länge anzuzeigen.

In Zeile 390 wird per INPUT ein Buchstaben L$ angefordert und eingegeben. Wird gleich das ganze Wort eingegeben und ist es richtig, springt die Zeile 400 auf den Abschnitt der Entscheidung über Spielwiederholung ab Zeile 600.

Zeile 410 ist die Sicherung gegen unbeabsichtigtes Drücken der RETURN-Taste ohne Eingabe.

Der schwierigste Teil des Programms ist der Vergleich des eingegebenen Buchstabens mit dem vorgegebenen Sportbegriff. Er beginnt ab Zeile 610.

Zeile 480 und 530 bilden eine Zählschleife in der Länge des vorgegebenen Sportbegriffs U$. Innerhalb dieser Schleife holt Zeile 490 die einzelnen Buchstaben aus dem Wort U$ heraus – mit der Methode des MID$-Befehls. Zeile 500 macht dasselbe im Gleichtakt für das (gepunktete) Hilfswort H$.

Zeile 500 vergleicht jetzt zuerst den eingegebenen – geratenen – Buchstaben L$ mit jedem einzelnen Buchstaben B$ des Wortes U$. Ist er identisch, wird ein neues Hilfswort N$ aus dem richtig geratenen Buchstaben L$ gebildet. War der Buchstabe falsch, dann wird aus dem Hilfswort H$ ein Punkt in das neue Hilfswort N$ übernommen.

Das wird für alle Stellen der gleichlangen Wörter U$ und H$ gemacht.

Danach wird in Zeile 540 das Hilfswort N$, das jetzt vielleicht schon eine Kombination aus Punkten und Buchstaben ist, dem alten Hilfswort H$ zugeordnet, welches ja dem Spieler beim nächsten Ratevorgang gezeigt wird (Zeile 370).

Danach wird der Ratevorgang in Zeile 390 und damit der ganze Ablauf wiederholt.

Jedesmal, wenn ein Buchstabe richtig geraten ist, kommt er an die entsprechende Stelle der Hilfswörter N$ und H$.

Erst wenn der Spieler das ganze Wort kennt und es komplett in Zeile 390 eingibt, springt, wie schon erwähnt, Zeile 400 nach Zeile 600, wo von 630 bis 670 in üblicher Manier die Frage nach Wiederholung des Spiels oder Ende gestellt wird.

Alle anderen Zeilennummern dienen der Lesbarkeit und der Schönschrift des Programms.

Das komplette Programm, mit Verzierungen zur besseren Lesbarkeit, ist in Listing 3 wiedergegeben.

```
10 PRINT CHR$(147)                            <Ø39>
2Ø PRINT"***** BEGRIFFE RATEN ******          <211>
3Ø PRINT                                      <132>
4Ø :                                          <Ø16>
5Ø REM-------- SPIELANLEITUNG----------       <2Ø6>
6Ø :                                          <Ø36>
7Ø :                                          <Ø46>
8Ø PRINT"DER COMPUTER DENKT SICH EINE         <155>
9Ø PRINT"SPORTART.                            <175>
1ØØ PRINT"SIE SOLLEN SIE ERRATEN.             <138>
11Ø PRINT                                     <212>
12Ø PRINT"SIE KOENNEN EINEN BUCHSTABEN        <Ø97>
13Ø PRINT"EINGEBEN, DANN ERSCHEINEN           <187>
14Ø PRINT"DIE GERATENEN BUCHSTABEN AN         <Ø76>
15Ø PRINT"DER RICHTIGEN STELLE.               <165>
16Ø PRINT                                     <ØØ6>
17Ø PRINT"HABEN SIE DAS WORT ERRATEN,         <Ø21>
18Ø PRINT"GEBEN SIE ES KOMPLETT EIN.          <169>
19Ø :                                         <166>
2ØØ :                                         <176>
21Ø REM------- WORT AUSWAEHLEN -------        <243>
22Ø :                                         <196>
23Ø :                                         <2Ø6>
24Ø FOR X=1 TO INT(RND(Ø)*14)+1               <155>
25Ø READ U$                                   <ØØ4>
26Ø NEXT                                      <Ø16>
27Ø :                                         <248>
28Ø H$=""                                     <Ø83>
29Ø FOR X=1 TO LEN(U$)                        <242>
3ØØ H$=H$+"."                                 <ØØ8>
31Ø NEXT                                      <Ø66>
32Ø :                                         <Ø42>
33Ø :                                         <Ø52>
34Ø REM-------- RATEVERSUCH ---------         <16Ø>
35Ø :                                         <Ø72>
36Ø PRINT:PRINT                               <Ø58>
37Ø PRINT"GESUCHT WIRD: ";H$                  <Ø73>
38Ø PRINT                                     <228>
39Ø INPUT"WAS RATEN SIE";L$                   <22Ø>
4ØØ IF L$=U$ THEN 6ØØ                         <117>
41Ø IF LEN(L$)>1 THEN 36Ø                     <Ø67>
42Ø :                                         <142>
43Ø :                                         <152>
44Ø REM--------- HILFSWORT ----------         <Ø65>
45Ø :                                         <172>
46Ø :                                         <182>
47Ø N$=""                                     <Ø41>
48Ø FOR X=1 TO LEN(U$)                        <176>
49Ø B$=MID$(U$,X,1)                           <196>
5ØØ C$=MID$(H$,X,1)                           <Ø76>
51Ø IF L$=B$ THEN N$=N$+L$:GOTO 53Ø           <Ø1Ø>
52Ø N$=N$+C$                                  <Ø67>
53Ø NEXT                                      <Ø32>
54Ø H$=N$                                     <Ø95>
55Ø GOTO 36Ø                                  <Ø82>
56Ø :                                         <Ø28>
57Ø :                                         <Ø38>
58Ø REM----  NOCH EINMAL ? ----------         <Ø43>
59Ø :                                         <Ø58>
6ØØ PRINT                                     <194>
61Ø PRINT"SIE HABEN RICHTIG GERATEN           <116>
62Ø PRINT                                     <214>
63Ø PRINT"NOCH EINMAL (J/N) ?                 <Ø96>
64Ø GET A$: IF A$="" THEN 64Ø                 <166>
65Ø IF A$="J" THEN RUN                        <166>
66Ø PRINT                                     <254>
67Ø PRINT"AUF WIEDERSEHEN                     <ØØ6>
68Ø :                                         <148>
69Ø :                                         <158>
7ØØ REM------- WOERTERSPEICHER--------        <211>
71Ø :                                         <178>
72Ø :                                         <188>
73Ø DATA FUSSBALL,HOCKEY,GOLF,BOXEN           <115>
74Ø DATA TURNEN,SCHWIMMEN,HOCHSPRUNG          <Ø99>
75Ø DATA SEGELN,FECHTEN,JUDO,BASKETBALL       <137>
76Ø DATA KEGELN,SCHIFAHREN,TENNIS             <1Ø2>
© 64'er
```

Listing 3. Ein kleines Spiel

LEKTION 19: Variable in Feldern (Arrays)

Ich lade Sie zu einem Experiment ein.

Wir stellen uns vor, wir benötigten in einem Programm eine Tabelle mit Zahlen, zum Beispiel die Anzahl der Bewohner, die in 10 einstöckigen Häusern in einer langen Straße wohnen. Die Adresse der Häuser unterscheidet sich nur durch die Hausnummer, denn der Straßenname bleibt ja gleich. Statt des langen Straßennamens nehmen wir nur einen Buchstaben, zum Beispiel T. Zur Kennzeichnung der 10 verschiedenen Adressen hängen wir die Ziffern 0 bis 9 hinten an. Die Zahl 10 können wir nicht nehmen, weil nach Punkt 3 des obigen Fazits die zweistelligen Variablennamen T1 und T10 identisch sind.

In unserem Beispiel wollen wir den einzelnen Häusern Bewohnerzahlen zuweisen, die jeweils um 2 größer sind als die Hausnummer.

```
10 T0=2
20 T1=3
30 T2=4
40 T3=5
etc.
100 T9=11
110 PRINT T0;T1;
etc. bis T9
```

Das ist natürlich eine sehr umständliche Arbeit, und langweilig ist diese dauernde Wiederholung obendrein.

Also, wie geht das einfacher und eleganter? Natürlich mit einer FOR-TO-NEXT-Schleife, aber aufgepaßt: bei ihrer Verwendung müssen wir die Hausnummer N in Klammern dem Straßennamen T anhängen. Anders akzeptiert das der C64 in Basic nicht.

Als Programm sieht das so aus:

```
10 FOR N=0 TO 9
20 T(N)=N+2
30 PRINT T(N)
40 NEXT N
```

Ihnen ist natürlich klar, daß Straßenname T plus Hausnummer (N) eine Variable ist, der wir Werte (Bewohner) zuweisen. Aber eins sollte Sie stutzig machen. Wie unterscheiden sich diese speziellen Variablen voneinander? Sind T(3) und T(8) nicht identisch, wenn nach der oben schon zitierten Regel nur die ersten beiden Zeichen des Namens – hier T(– hergenommen werden ?

Nun, die Klammer macht den Unterschied. Durch sie wird ein neuer Typ einer Variablen, eine sogenannte Feld-Variable definiert, und Feld-Variablen unterscheiden sich durch die Zahl innerhalb der Klammer. Diese Zahl heißt Index (Mehrzahl: Indizes). Warum sie ausgerechnet Feld-Variable heißt, erkläre ich gleich.

Vorher, bei der Schreibweise T0, T1 etc. waren wir auf 10 Variable beschränkt, weil T1 und T10 dieselbe Variable war. Jetzt – mit Indizes als Unterscheidungmerkmal – gilt diese Beschränkung nicht. Deshalb wollen wir die Anzahl der Indizes im Programm noch erhöhen. Bei einem N von 0 bis 10 geht es ohne Probleme. Aber ab 11 ist schon wieder Feierabend. Wir erhalten die Fehlermeldung »BAD SUBSCRIPT ERROR IN 20«. Um Ihnen das zu erklären, muß ich genauer beschreiben, was passiert, wenn wir eine Variable mit einem Index in Klammern verwenden.

Wenn wir eine Variable so schreiben:
T(1)=25
dann nimmt der Computer an, daß wir in einer Tabelle noch mehrere derartige Variable T( ) verwenden wollen – klar, sonst würden wir uns ja die Mühe mit der Klammer nicht machen. Um uns die Sache zu erleichtern, reserviert der Computer unter dem Variablennamen T in seinem Speicher von vornherein 11 Plätze, für T(0) bis T(10). Diesen reservierten Bereich können Sie salopp Tabelle nennen; offiziell heißt er *Feld* oder auf englisch *Array*. Die alte Regel für den Namen der Variablen gilt jetzt auch nicht mehr, denn zum Unterscheiden der einzelnen Feld-Variablen desselben Anfangsbuchstabens bedient der Computer sich der Index-Zahl in der Klammer.

Die Beschränkung auf eine Feld-(Array-)Größe von 11 Plätzen wäre natürlich sehr lästig, wenn sie nicht umgangen werden könnte. Wenn wir nämlich mehr Platz brauchen, können wir dem Computer mit dem Basic-Befehl
DIM
unsere Reservierungswünsche mitteilen. DIM ist eine Abkürzung, die aus dem Wort »Dimension« abgeleitet ist. Als Beispiel möge die folgende Programmzeile dienen:

```
→ 5 DIM T(25)
```

Sie reserviert im obigen Programm für die Variable T() im Speicher ein Feld von 26 Plätzen, von T(0) bis T(25).

Die Größe eines Feldes ist nur durch den vorhandenen Speicherplatz begrenzt. Wenn Sie die Zahl zu groß wählen, verweigert der Computer die Reservierung – natürlich erst nach dem Befehl RUN – mit OUT OF MEMORY, was soviel heißt wie »keinen Platz mehr im Speicher«.

Basic-Befehl Nr. 26 DIM
- dieser Befehl wird in der folgenden Schreibweise verwendet: DIM Variablenname (Index)
- er reserviert einen Speicherbereich für eine durch den Index festgelegte Anzahl von Variablen desselben Namens, die sich nur durch ihren jeweiligen Index unterscheiden. Dieser Speicherbereich wird Feld oder Array genannt
- mit DIM können sowohl Felder für numerische, als auch für String-Variable reserviert werden. Ein Feld kann immer nur einen einzigen Variablentyp enthalten
- für den Namen und für die Kennzeichnung des Typs der Feld-Variablen (vor der Klammer) gelten dieselben Regeln wie für alle anderen Variablen

Ein Feld kann also auch aus String-Variablen bestehen:

```
→ 200 DIM A$(25)
  210 A$(0)="FEUER"
  220 A$(1)="ZANGEN"
  230 A$(2)="BOWLE"
       .   .
       .   .
       .   .
  300 FOR N=0 TO 2
  310 PRINT A$(N);
  320 NEXT N
```

Dieses kleine Programm reserviert ein Feld von 25 Strings und weist den ersten dreien davon je ein Wort zu. Mit RUN 200 gestartet, druckt es das Wort Feuerzangenbowle aus, indem in der Zeile 310 nacheinander für die Werte von N=0 bis N=2 die gespeicherten Strings A$(0) bis A$(2) durch das Semikolon nach dem PRINT-Befehl aneinandergeklebt werden. Das Eintragen der einzelnen Werte in die Tabelle, das ich in den Zeilen 210 bis 230 vorgenommen und danach nur noch angedeutet habe, geht viel eleganter mit den Befehlen READ-DATA, die in der Lektion 17 beschrieben sind.

Um das zu zeigen, schlage ich Ihnen eine kleine Anwendung – im Computerdeutsch heißt so ein Programm »Utility« – vor, und zwar eine Nachschlagliste von Geburtstagen Ihrer Freunde und Verwandten. Natürlich ist das keine Utility, deretwegen ich mir einen Computer kaufen würde, aber für unsere Zwecke hier ist sie ganz passend.

Ich lege das Beispiel vorerst einmal auf 5 Namen (N$) und dazugehörige Geburtstagsdaten (D$) aus. Wir brauchen also 2 Felder mit je 5 Plätzen, dazu zwei DATA-Zeilen mit den Eintragungen. Verwenden Sie bitte meine wirr erscheinenden Zeilennummern; am Ende passen sie schon zusammen.
Speicher löschen mit NEW

```
→ 20 DIM N$(4)
  500 DATA MAX,MORITZ,MARIA,HANS,LUISE
  120 DIM D$(4)
  600 DATA 12.6.52,3.4.60,21.1.40,19.9.56,11.11.
  70
```

Denken Sie bitte daran: DIM N$(4) legt 5 Plätze fest, nämlich 0 bis 4.

Denken Sie ebenfalls daran, daß Eintragungen in DATA-Zeilen durch ein Komma voneinander getrennt sein müssen.

So, jetzt brauchen wir für beide Felder eine Schleife, mit der die Eintragungen der DATA-Zeilen mit READ in das jeweilige Feld gelesen werden. Für das Namensfeld gilt:

```
→ 30 FOR I=0 TO 4
  40 READ N$(I)
  50 NEXT I
```

Dasselbe machen wir für das zweite Feld der Geburtstage:

```
→ 130 FOR I=0 TO 4
  140 READ D$(I)
  150 NEXT I
```

Jetzt brauchen wir noch einen Programmteil, der nach Eingabe eines Namens das dazugehörige Geburtsdatum heraussucht und ausdruckt. Für die »Dazugehörigkeit« benützen wir die Indizes der Feldvariablen. Das bedeutet nichts anderes, als daß zum zweiten Namen N$(2) das zweite Datum D$(2) gehört.

Doch zuerst stellen wir per INPUT die Frage F$ nach dem Namen, dessen Geburtstag wir wissen wollen:

```
→ 200 INPUT "NAME";F$
```

Dann vergleichen wir in einer weiteren Schleife die gespeicherten Namen N$(I) mit dem gefragten Namen F$. Wenn bei einem bestimmten Wert I der Vergleichsschleife die beiden Namen gleich sind, wird der Geburtstag mit demselben Index I ausgedruckt. Das Programm kann dann eine weitere Eingabe eines Namens verlangen (GOTO 200).

```
→ 210 FOR I=0 TO 4
  220 IF N$(I)=F$ THEN PRINT D$(I):GOTO 200
  230 NEXT I
```

Um den Vergleichs- und Entscheidungsvorgang genau zu verstehen, brauchen Sie lediglich die Schleifen durchzuspielen, indem Sie die Werte von I gedanklich hochzählen und im Programm nachschauen, was passiert. Wir können aber zum Verständnis auch eine »Lehrzeile« (mit h !) einfügen, die uns den jeweiligen Stand des Vergleiches anzeigt:

```
→ 215 PRINT I;F$,N$(I);D$(I)
```

Nach RUN und nach Eingabe eines der 5 Namen sehen wir am Bildschirm eine Reihe mit steigenden Werten von I, daneben (Semikolon !) den eingegebenen Namen, dann in einem größeren Abstand (Komma !) die Namen und Daten, und zwar so lange, bis F$ und N$ gleich sind.

Nehmen Sie jetzt Zeile 215 wieder heraus, aber fügen Sie bitte eine Zeile 240 dazu, die wir für ein bedienungsfreundliches Programm brauchen. Wir müssen nämlich Vorkehrung treffen für den Fall, daß ein Name eingegeben wird, der nicht in der Liste steht. Auch Tippfehler fallen in diese Kategorie. Wenn kein positiver Vergleich zwischen F$ und N$ innerhalb der Schleife auftritt, macht das Programm nach der Schleife weiter, mit der neuen Zeile 240, die für sich selbst spricht:

```
→ 215
  240 PRINT"NAME IST NICHT IN DER LISTE"
```

Das ganze Programm sieht so aus:

```
→ 20 DIM N$(4)
  30 FOR I=0 TO 4
  40 READ N$(I)
  50 NEXT I
  120 DIM D$(4)
  130 FOR I=0 TO 4
  140 READ D$(I)
  150 NEXT I
  200 INPUT "NAME";F$
  210 FOR I=0 TO 4
  220 IF N$(I)=F$ THEN PRINT D$(I):GOTO 200
  230 NEXT I
  240 PRINT"NAME IST NICHT IN DER LISTE"
  500 DATA MAX,MORITZ,MARIA,HANS,LUISE
  600 DATA 12.6.52,3.4.60,21.1.40,19.9.56,11.11
  .70
```

Der DIM-Befehl und der READ-Befehl haben beide eine zusätzliche Eigenschaft, die uns eine wesentliche Verbesserung dieses Programms erlaubt:

Man darf hinter einem einzigen DIM-Befehl mehrere Felder dimensionieren. Sie müssen lediglich durch ein Komma getrennt sein. Genauso darf man mit einem einzigen READ-Befehl mehrere Werte-Gruppen auslesen. Hier gilt dieselbe Kommaregel wie bei DIM.

Damit können wir die Zeilen 20 bis 150 stark verkürzen:

```
→ 20 DIM N$(4),D$(4)
  30 FOR I=0 TO 4
  40 READ N$(I),D$(I)
  50 NEXT I
```

Die Zeile 20 ist einfach nur eine Zusammenziehung der beiden alten Zeilen 20 und 120. Die Funktion des DIM-Befehls bleibt dabei erhalten – er dimensioniert halt gleich beide Felder.

Beim »doppelten« READ-Befehl in Zeile 40 hat sich in der Arbeitsweise, verglichen mit den alten Zeilen 40 und 140, etwas geändert. Schuld daran ist aber die FOR-NEXT-Schleife der Zeile 30.

Bei jedem Wert von I holt nämlich der READ-Befehl zwei hintereinanderliegende Werte aus den DATA-Zeilen und weist sie den beiden Variablen N$ und D$ zu. Wir müssen diesem Vorgang dadurch Rechnung tragen, daß wir den Inhalt der DATA-Zeilen umorganisieren. Es müssen jetzt immer je ein Name und das dazugehörige Geburtsdatum hintereinander kommen, was eigentlich viel leichter einzutippen ist. Sie sehen, gute Programmierung ist fast immer auch klarer und logischer.

```
→ 500 DATA MAX,31.3.55,MORITZ,12.4.45,
  HANS,6.2.60
  600 DATA MARIA,14.7.63,LUISE,8.9.60
```

Mit dieser Anordnung haben wir noch einen weiteren Vorteil erhalten. Das Programm, oder besser gesagt die Geburtstagskartei, kann ganz leicht erweitert werden. Einen neuen Namen samt Datum brauchen Sie nur durch Komma getrennt hinter die letzte Eintragung der DATA-Zeilen zu schreiben.

Doch halt! Noch etwas fehlt: Die Zahl der Namen, die sich durch eine neue Eintragung natürlich erhöht, kommt ja im Programm auch vor, und zwar in unserem Fall als Ziffer 4 in den Zeilen 20, 30 und 210. Um uns diese Zusatzarbeit ebenfalls zu erleichtern, geben wir dieser Zahl einen Variablen-Namen Z, definieren dieses Z ganz am Anfang und brauchen so bei jeder neuen Namenseintragung nur dieses Z um 1 erhöhen.

Diese Zeilen sehen jetzt so aus:

```
→ 10 Z=4
  20 DIM N$(Z),D$(Z)
  30 FOR I=0 TO Z
  210 FOR I=0 TO Z
```

FAZIT

1. Hinter einem DIM-Befehl können mehrere Felder dimensioniert werden. Sie müssen lediglich durch ein Komma getrennt sein.
2. Mit einem READ-Befehl können mehrere DATA-Werte gelesen werden. Auch diese müssen durch ein Komma voneinander getrennt werden.
3. Es empfiehlt sich, einen Zahlenwert oder einen String, der mehrmals in einem Programm vorkommt, ganz am Anfang des Programms als numerische beziehungsweise als Stringvariable zu definieren und im Programm dann nur einmal dieser Variablen den Wert zuzuweisen.

Wir haben jetzt ein komplettes kleines Programm, dessen zusammengewürfelte Zeilennummern ihm allerdings ein unfertiges Aussehen verleihen.

Deswegen und wegen der vielen Änderungen während seiner Entstehung ist es im folgenden Listing 4 noch einmal komplett dargestellt.

```
10 Z=4                                        <075>
20 DIM N$(Z),D$(Z)                            <096>
30 FOR I=0 TO Z                               <245>
40 READ N$(I),D$(I)                           <035>
50 NEXT I                                     <134>
200 INPUT "NAME";F$                           <194>
210 FOR I=0 TO Z                              <169>
220 IF N$(I)=F$ THEN PRINT D$(I):GOTO 200     <246>
230 NEXT I                                    <058>
240 PRINT "NAME IST NICHT IN DER LISTE        <249>
500 DATA MAX,31.3.55,MORITZ,12.4.45,HANS,6
 .2.60                                        <186>
600 DATA MARIA,14.7.63,LUISE,8.9.60           <196>
```


Listing 4. Eine einfache Geburtstagsverwaltung

LEKTION 20: Zweidimensionale Felder (Arrays)

Hinter dieser Überschrift verbirgt sich eine Erweiterung des DIM-Befehls.

- eindimensional ist eine Linie; sie hat nur in einer Richtung eine Ausdehnung
- zweidimensional ist eine Fläche; sie hat eine Länge und eine Breite.

Diese Betrachtungsweise kann auch auf Variablenfelder angewendet werden.

Ich habe bei der ersten Erklärung des Begriffs »Feld« und »Feldvariable« den Vergleich mit einstöckigen Häusern in

einer Straße verwendet. Unser damaliges Beispiel könnte man in dieser Form aufmalen:

Bewohner	2	3	4	5	6
Adresse	T(0)	T(1)	T(2)	T(3)	T(4)

Ein zweidimensionales Feld ist eine Tabelle, in der sowohl waagerecht wie senkrecht Eintragungen möglich sind:

In unserem Beispiel sind das Häuser, die mehrere Stockwerke und pro Stockwerk verschiedene Bewohnerzahlen haben. Der Einfachheit halber gebe ich allen Häusern dieselbe Zahl von Stockwerken.

Stockwerk 3	T(0,3)	T(1,3)	T(2,3)	T(3,3)	T(4,3)
2	T(0,2)	T(1,2)	T(2,2)	T(3,2)	T(4,2)
1	T(0,1)	T(1,1)	T(2,1)	T(3,1)	T(4,1)
0	T(0,0)	T(1,0)	T(2,0)	T(3,0)	T(4,0)
Hausnummer	0	1	2	3	4

Bei diesen Reihenhäusern habe ich nicht mehr die Zahl der Bewohner in die Häuser beziehungsweise Stockwerke geschrieben, sondern die »Adressen«, die wieder unsere Feldvariablen sind. Nur haben sie diesmal zwei Indizes, einen für das Stockwerk (Zeile des Feldes) und einen für die Hausnummer (Spalte des Feldes).

Wenn wir jetzt sagen wollen, daß im 3. Stock des 1. Hauses 7 Leute wohnen, legen wir das fest mit:

```
T(1,3)=7
```

Bei der Definition einer zweidimensionalen Variablen T(A,B) reserviert auch diesmal wieder der Computer ein Feld von 11 Variablen, allerdings pro Index, das heißt insgesamt 11 * 11=121 Plätze.

Ein kleines Programm beweist diese Behauptung:

```
10 FOR I=0 TO 10
20  FOR K=0 TO 10
30   T(I,K)=I+K
40   PRINT T(I,K);
50  NEXT K
60 NEXT I
```

Ich habe es extra so geschrieben, daß Sie die beiden Schleifen besser sehen können. Pro Durchlauf der äußeren I-Schleife läuft die innere K-Schleife 11mal durch. Dementsprechend sieht das Resultat der Zeile 40 aus.

Wenn Sie jetzt die obere Grenze von I auf 11 erhöhen, merkt das Programm erst nach dem 121.Durchlauf, daß zuwenig Platz reserviert worden ist. Lassen Sie dagegen das I auf maximal 10, erhöhen aber das K auf 11, bleibt das Programm schon nach dem ersten Durchlauf der inneren Schleife stehen.

Das war eine kleine Erinnerung an die Wirkungsweise geschachtelter Schleifen.

Wenn wir ein größeres Feld benötigen, müssen wir diesen Bedarf wieder mit dem DIM-Befehl eingeben:

DIM T(25,34)

FAZIT

1. Felder können auch mehrere Dimensionen haben. Eine zweidimensionale Feld-Variable hat 2 Indizes in der Klammer, die durch ein Komma getrennt sein müssen.
2. Für eine mehrdimensionale Feld-Variable werden pro Index 11 Plätze im Speicher reserviert.
3. Zur Dimensionierung größerer Felder steht der DIM-Befehl, ebenfalls mit 2 Indizes, zur Verfügung.
4. Für mehrdimensionale DIM-Befehle und Feld-Variable gelten dieselben Regeln wie für eindimensionale Felder.

Um die Wirkungsweise eines zweidimensionalen Feldes vorzuführen, habe ich vor, mit Ihnen einen »Karteikasten« zu entwickeln, in dem Sie eine Literatursammlung, Geschichtsdaten, Personalunterlagen oder alle Titel Ihrer Plattensammlung katalogisieren können. Eine Kartei ist natürlich nur dann sinnvoll, wenn bestimmte Eintragungen schnell durch Angabe eines Stichwortes gefunden werden können.

Ich selbst habe mir für meine Arbeiten eine Computer-Literatursammlung zugelegt, deren Struktur ich hier verwenden will. Von jedem interessanten Artikel oder Buchkapitel mache ich folgende Eintragung:
- Zeitschrift (oder Buch)
- Titel des Aufsatzes
- Sachgebiet
- Datum der Veröffentlichung
- Seitennummer
- Computer-Typ

Diese Eintragungen nenne ich DATENSATZ, jeder Datensatz enthält 6 KATEGORIEN.

Diese sollen nun in einem zweidimensionalen Feld untergebracht werden.

KATEGORIE	K=1	K=2	K=3	K=4	K=5	K=6
DATENSATZ	ZEITSCHRIFT	TITEL	SACHGEBIET	DATUM	SEITE	COMPUTERTYP
D=1	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)
D=2	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)
D=3	(3,1)	(3,3)	usw.			

Der erste Index »D« gibt also die Nummer des Datensatzes an, der zweite Index »K« die jeweilige Kategorie.

Um fürs erste einmal 100 Datensätze zu je 6 Kategorien eintragen zu können, dimensionieren wir ein Feld A$:

```
110 DIM A$(100,6)
```

Die ersten 5 Datensätze könnten zum Beispiel so aussehen:

```
1002 DATA GAZETTE,SPEEDSCRIPT,UTILITY,
1/84,12,64
1003 DATA RUN,DER DIM-BEFEHL,BASIC,3/85,125,20
1004 DATA 64ER,FILES,KURS,4/87,118,64
1005 DATA CHIP,SCHLEIFEN,BASIC,11/83,134,20/64
1006 DATA HAPPY,GARBAGE COLLECTION,KURS,
4/87,20/64
```

Diese Datensätze in DATA-Zeilen lesen wir mit den folgenden Zeilen in das Feld A$:

```
120 D=D+1
130 FOR K=1 TO 6
140 READ A$(D,K)
150
160 NEXT K
170 GOTO 120
```

Da wir nicht wissen können, wieviele Datensätze wirklich im Feld enthalten sind, muß die D-Schleife hochgezählt werden, während die K-Schleife fest von 1 bis 6 läuft, da ja die Anzahl der Kategorien mit 6 festliegt.

Nun brauchen wir am Ende der Datensätze - ich lege es auf Zeile 2000 - eine Endmarkierung; sonst läuft der READ-Vorgang über. Ich wähle dafür den Klammeraffen »@«, dessen Auftreten wir in Zeile 150 abfragen:

```
150 IF A$(D,K)="@" GOTO 210
2000 DATA @
```

Ab Zeile 210 soll der Suchvorgang nach einem bestimmten Datensatz beginnen. Der Suchvorgang beginnt mit der Eingabe des Stichwortes S$. Es kann ein Sachgebiet sein,

eine Zeitschrift oder ein bestimmter Titel, den Sie suchen. Auf alle Fälle soll das Programm alle Datensätze, die das Stichwort enthalten, ausdrucken.

```
210 INPUT"STICHWORT";S$
220 FOR S=1 TO D
230 FOR K=1 TO 6
240 IF A$(S,K)=S$ THEN FOR Z=1 TO 6:PRINT A$
    (S,Z):NEXT Z
250 NEXT K
260
270 NEXT S
280 PRINT "ENDE DER SUCHE"
```

Um alle Datensätze, die im Feld A$ stehen, nach dem Stichwort S$ abzusuchen, bilden wir zwei verschachtelte Schleifen. Die eine zählt die Datensätze von 1 bis zum letzten Datensatz, dessen Nummer vom Einlesevorgang her noch auf dem Wert D steht. Für diese Schleife wähle ich die Variable S. Die andere Schleife zählt wieder die Kategorien K von 1 bis 6 (Zeilen 220 und 230).

In Zeile 240 wird pro Schritt geprüft, ob die Feldeintragung mit dem Stichwort S$ übereinstimmt.

Stimmt sie überein, dann soll der ganze Datensatz, in dem das Stichwort gefunden worden ist, ausgedruckt werden. Deswegen legen wir hinter das THEN sofort eine Sechserschleife zum Ausdrucken aller 6 Kategorien.

Danach wird die Suche mit den K- und S-Schleifen fortgesetzt, denn das Stichwort kann ja auch in nachfolgenden Datensätzen vorkommen.

Wenn Sie dieses Programmfragment laufen lassen, wird es aussteigen mit der Fehlermeldung »NEXT WITHOUT FOR«.

Der Fehler, der hier auftritt, ist so häufig und doch so schwer zu sehen, daß ich ihn extra hier eingebaut habe, um Sie daraufzustoßen.

Wir haben in den Zeilen 120 bis 150 eine Schleife mit der Variablen K für die 6 Kategorien. Dasselbe wiederholen wir in den Zeilen 230 bis 250. Und genau das geht schief, weil wir nämlich aus der ersten K-Schleife in Zeile 150 herausspringen, ohne daß die Schleife zu Ende gelaufen ist. Im Speicher des Computers steht also noch eine Schleifenvariable K auf einem bestimmten Wert, wenn wir in Zeile 230 eine neue K-Schleife anfangen. Das kann der Computer nicht verarbeiten.

Wir zählen übrigens die 6 Kategorien noch einmal in der Zeile 240, nur da macht es nichts, denn wir haben eine andere Schleifenvariable, nämlich Z verwendet.

Das ist also schon eine der möglichen Lösungen, nämlich eine Schleifenvariable immer nur einmal zu verwenden. Die sauberste Lösung ist aber, nach dem Aussprung die Schleife »künstlich« zum Ende zu bringen und das wollen wir in Zeile 150 auch machen. Verbessern Sie bitte:

```
150 IF A$(D,K)="@"THEN K=6:NEXT K:GOTO 210
```

Wir setzten also die Schleifenvariable K auf ihren Endwert 6 und geben ein letztes NEXT K. Damit wird die Schleife »geschlossen« und wir dürfen ungestraft K als Schleifenvariable wiederverwenden.

Jetzt bleibt nur noch die Frage an den Benutzer, ob der Programmlauf wiederholt oder abgebrochen werden soll (Zeilen 310 bis 340). Das komplette Programm steht in Listing 5.

Diese »Kartei« kann durch Eintippen von zusätzlichen DATA-Zeilen beliebig vergrößert werden. Soll sie mehr als 100 Datensätze enthalten, dann muß in Zeile 110 das Feld entsprechend vergrößert dimensioniert werden. Letztlich ist das Programm nur begrenzt durch den verfügbaren Speicher des C64.

Das zweidimensionale Feld mit Variablen steht also im Speicher des C64 gespeichert. Wenn wir den Computer ausschalten, wird der Speicher gelöscht und alles ist weg.

```
100 REM**** DATEN INS FELD LESEN ****        <071>
105 :                                        <081>
110 DIM A$(100,6)                            <230>
120 D=D+1                                    <003>
130 FOR K=1 TO 6                             <119>
140 READ A$(D,K)                             <191>
150 IF A$(D,K)="@"THEN K=6:NEXT K:GOTO 210   <171>
160 NEXT K                                   <004>
170 GOTO 120                                 <130>
195 :                                        <171>
200 REM**** SUCHVORGANG *****                <020>
205 :                                        <181>
210 INPUT"STICHWORT";S$                      <148>
220 FOR S=1 TO D                             <024>
230 FOR K=1 TO 6                             <219>
240 IF A$(S,K)=S$ THEN FOR Z=1 TO 6:PRINT
    A$(S,Z):NEXT Z                           <162>
250 NEXT K                                   <094>
260 PRINT                                    <108>
270 NEXT S                                   <180>
280 PRINT"ENDE DER SUCHE"                    <165>
295 :                                        <017>
300 REM**** WIEDERHOLUNG ODER ENDE ****      <101>
305 :                                        <027>
310 PRINT"NOCH EINMAL (J/N)?"                <064>
320 GET V$:IF V$="" THEN 320                 <182>
330 IF V$="J" THEN 210                       <054>
340 END                                      <088>
999 :                                        <213>
1000 REM**** DATENSAETZE *****               <166>
1001 :                                       <215>
1002 DATA GAZETTE,SPEEDSCRIPT,UTILITY,1/84
     ,12,64                                  <001>
1003 DATA RUN,DIM,BASIC,3/58,125,20          <044>
1004 DATA 64ER,FILES,KURS,4/87,118,64        <145>
1005 DATA CHIP,SCHLEIFEN,BASIC,11/83,134,2
     0/64                                    <066>
1006 DATA HAPPY,GARBAGE COLLECTION,KURS,4/
     87,38,20/64                             <082>
10000 DATA @                                 <110>
```

© 64'er

Listing 5. Zweidimensionales Feld

Es wäre deshalb schön, wenn wir das Feld mit den Daten auch auf ein Band oder auf eine Diskette abspeichern könnten, um es bei einer späteren Gelegenheit wieder in den Computer holen zu können.

Aus dem Betriebshandbuch von Commodore kennen Sie sicher die Speicher- und Ladebefehle SAVE, LOAD für Kassetten und Disketten. Ich sage es aber besser gleich, bevor Sie unnütz experimentieren: Diese Befehle speichern und laden leider nur Programme, nicht aber Variablenwerte, die im Lauf eines Programms erst erzeugt werden.

Es gibt aber noch eine andere Methode, wie man Variablenwerte, sei es für numerische, sei es für Feldvariable, speichern kann. Dazu aber muß ich Ihnen erst in Lektion 21 ein bißchen mehr über den Speicher des C64 erklären. Speichern Sie bitte das Fragment Listing 5 auf Band oder Diskette ab, wir werden es später vervollständigen.

FAZIT

1. Schleifen, die durch Aussprung verlassen werden, bevor sie zu Ende gelaufen sind, werden im Speicher des Computers als »noch offen« registriert. Ihre Schleifenvariable kann daher nicht wiederverwendet werden.
2. Entweder muß man die Wiederverwendung von Schleifenvariablen vermeiden oder muß verlassene Schleifen schließen durch Zuweisung der Variablen auf den vorgesehenen Endwert plus nachfolgendem NEXT-Befehl.
3. Zweidimensionale Felder werden durch zwei verschachtelte Schleifen abgearbeitet.
4. Felder beziehungsweise die Werte der Feldvariablen können nicht mit SAVE auf Kassette oder Diskette gespeichert werden.

Der nun folgende Abschnitt hat nur indirekt etwas mit Basic zu tun. Aber wie ganz am Anfang betont, komme ich ohne einige Beschreibungen von Eigenschaften oder besser gesagt von Eigenheiten des C64 nicht aus. Einige Kenntnisse des Innenlebens des Computers gehören halt doch auch zum Programmieren.

FAZIT:
1. Schleifen, die durch Aussprung verlassen werden, bevor sie zu Ende durchgelaufen sind, werden im Speicher des Computers als »noch offen« registriert. Ihre Schleifenvariablen können daher nicht weiterverwendet werden.
2. Entweder muß man die Wiederverwendung von Schleifenvariablen vermeiden, oder muß verlassene Schleifen schließen durch Zuweisung der Variable auf den vorgesehenen Endwert des nachfolgenden NEXT-Befehls.
3. Zweidimensionale Felder werden durch zwei ineinandergeschachtelte Schleifen abgearbeitet.
4. Feldvariable können nicht mit SAVE auf Kassette oder Diskette gespeichert werden.

Unsere Kartei konnte durch Einfügen von zusätzlichen DATA-Zeilen beliebig vergrößert werden. Soll es mehr als 100 Datensätze enthalten, dann muß in Zeile 110 das Feld entsprechend größer dimensioniert werden. Letztlich ist das Progamm nur begrenzt durch den Speicherplatz im C64.

LEKTION 21: Der Speicher – das Gedächtnis des Computers

Alle Computer – Großrechenanlagen genauso wie kleine Heimcomputer – sind aus den folgenden Grundbausteinen aufgebaut:
- zentrale Recheneinheit, auch Mikroprozessor oder CPU (Central Prozessing Unit) genannt
- Speichereinheit
- Ein- und Ausgabe-Bausteine

Während die CPU rechnet, alle Vorgänge im Computer steuert, Befehle ausführt und somit das eigentliche Herz des Computers darstellt, braucht man die Ein- und Ausgabe-Bausteine, um Daten in den Computer hineinbeziehungsweise herauszuleiten. Alle Anschlüsse von Datasette, Diskettenlaufwerk, Drucker und Bildschirm werden von ihnen gesteuert.

Der Speicher ist der Notizblock des Computers. In ihm steht alles, was er sich merken soll: Programmzeilen, Variablenwerte, Zeichenketten (Strings) und so weiter. Auf englisch heißt Speicher passenderweise Memory, was wir wiederum auf deutsch mit Gedächtnis übersetzen könnten.

Jeder Computer hat zwei Arten von Speicher:

Das RAM (Random Access Memory) ist ein aus elektronischen Bauteilen aufgebauter Speicher, der für Programme frei verfügbar ist. Wir können in diesen Speichertyp Daten hineinschreiben und sie wieder herauslesen, ohne sie zu zerstören. Nur nach dem Ausschalten des Computers – da sind sie weg!

Das passiert auch dann, wenn nur kurzzeitig der Strom ausfällt, etwa bei einem Gewitter oder wenn der Netzstecker einen Wackelkontakt hat.

Es ist deshalb ratsam, bei längeren Programmierarbeiten Zwischenergebnisse, das heißt den jeweiligen Inhalt des RAM auf Kassette oder Diskette abzuspeichern, denn nur dort sind sie dauerhaft sicher.

Das ROM (Read Only Memory) ist nicht frei verfügbar. Es besteht zwar auch aus elektronischen Bauteilen, aber sein Inhalt ist fest »eingebrannt« – man kann ihn nicht ändern. Er wird vom Computer selbst verwendet. Wenn Sie zum Beispiel Ihren C64 einschalten, dann läuft eine im ROM eingespeicherte Folge von Programmschritten ab, die schließlich mit der Meldung des Computers auf dem Bildschirm endet, mit der er sich bereit (READY) meldet. Im ROM stehen nicht nur die Programmschritte, die den Betrieb des Computers steuern, sondern auch das Programm, welches alle Basic-Befehle in einen Code »übersetzt«, den der zentrale Mikroprozessor (CPU) versteht. Wenn Sie mehr über die allgemeine Arbeitsweise des Computers erfahren wollen, dann lesen Sie bitte im 64er Sonderheft 16 ab Seite 16 nach.

21.1. Die Adressen der Speicherzellen

Der gesamte Speicher des C64 ist aus einzelnen Speicherzellen aufgebaut. Der C64 bietet Platz für 65536 Speicherzellen. Jede dieser Speicherzellen hat eine Nummer, von 0 bis 65535.

Diese Nummern nennen wir Adressen. Um eine Zahl oder ein Zeichen in eine bestimmte Speicherzelle hineinschreiben zu können, müssen wir ihre Adresse kennen. Natürlich gilt dasselbe für das Auslesen. Es liegt deshalb sehr nahe, uns ein »Adreßbuch« des Speichers zu beschaffen.

Im Betriebshandbuch ist eine derartige Liste im Anhang auf den Seiten 160 bis 165 unter dem Titel »Speicherbelegung« angegeben. Diese Liste ist aber alles andere als klar und verständlich. In dem folgenden Absatz habe ich daher eine andere Darstellung gewählt, die Ihnen eine Übersicht über die verschiedenen Speicherbereiche und ihre Bedeutung geben soll. Aber ein Adreßbuch, oder wie es auf englisch heißt, eine »Memory Map« ist das eigentlich auch nicht. Dazu müßte ich ja jede einzelne Adresse beschreiben.

Hier ist also eine Kurzfassung eines Adreßbuches, in der wir aber bereits viel sehen können:
- Der Speicher beginnt ganz unten bei Adresse 0.
- Die ersten 2047 RAM-Speicherzellen sind vom Computer selbst belegt.
- Ab Adresse 2048 beginnt der RAM-Speicher für Basic-Programme. Ab hier werden alle eingegebenen oder von Band und Diskette geladenen Programme gespeichert. Auch alle Variablen, Felder und Zeichenketten, die im Lauf eines Programmes auftauchen, werden dort gespeichert.
- Dieser Speicherbereich endet bei Adresse 40959.

Dem Programmierer stehen 38911 Speicherplätze zur Verfügung. Alle diese Speicherzellen sind also RAM-Zellen, das heißt, man kann Daten hineinschreiben und herauslesen.
- Von 40960 bis 49151 sind die fest vorgegebenen Programme zum Übersetzen von Basic in den Maschinencode enthalten
- Anschließend an 53248 bis 57343 sind alle Buchstaben und Zeichen und die Ein- und Ausgabeprogramme gespeichert.
- Danach folgen die Programme des Betriebssystems, auch Kernel genannt.

Bemerkenswert ist, daß die Betriebsprogramme des Computers (Basic-Übersetzer, Zeichensatz, Ein- und Ausgabe und Kernel) in einem ROM-Speicher untergebracht sind, der über einem RAM-Speicher sitzt. Das heißt, sie haben dieselben Adressen. Wie dieser RAM-Teil des Speichers genutzt werden kann, gehört zur höheren Programmierkunst und wird hier nicht behandelt.

Uns stellt sich als nächstes die Frage, wie wir Daten in einzelne Speicherzellen des Programmspeichers hineinschreiben und herauslesen können.



21.2. Im Speicher stöbern

Basic hat zwei Befehle, mit deren Hilfe der Speicher abgefragt werden kann.

Während der Lese-Befehl völlig ungefährlich ist – er »kopiert« praktisch den Inhalt der Speicherzelle und läßt das Original unverändert – kann der Schreib-Befehl gefährlich sein. Ich will Sie gleich zu Beginn davor warnen, ihn unbedacht einzusetzen – er verändert den Inhalt einer Speicherzelle, und über die Folgen dieser Veränderung für ein Programm oder für den Betrieb des Computers muß man sich im klaren sein.

Der Schreib-Befehl heißt

POKE

was das englische Wort für »hineinstochern« ist (nomen est omen). Nach dem Befehlswort muß die Adresse der Speicherzelle stehen, danach folgt, durch ein Komma getrennt, die Zahl, die hineingeschrieben werden soll.

Der Befehl

```
POKE 14000,55
```

schreibt also die Zahl 55 in die Speicherzelle 14000.

Basic-Befehl Nr. 27 POKE

- er wird in der folgenden Weise geschrieben:
  POKE Adresse,Wert

Adresse und Wert müssen durch ein Komma voneinander getrennt sein

- der POKE-Befehl speichert den »Wert« in der mit »Adresse« bezeichneten Speicherzelle ab. Er überschreibt dabei einen dort gespeicherten früheren Wert
- gültige »Werte« liegen im Bereich von minimal 0 bis maximal 255. Wird dieser Bereich überschritten, erscheint die Fehlermeldung ILLEGAL QUANTITY ERROR.
- der erlaubte Bereich für »Adresse« reicht von 0 bis 65535. Ein Überschreiten wird mit der Fehlermeldung ILLEGAL QUANTITY ERROR bestraft.

Der Lese-Befehl heißt

PEEK

das ist das englische Wort für »hineinschauen«. Hinter diesem Befehlswort steht die Adresse – und zwar in Klammern! – deren Inhalt gelesen werden soll.

Die Befehlsfolge:

```
A=PEEK(14000):PRINT A
```

oder noch kürzer

```
PRINT PEEK(14000)
```

liest den Inhalt der Speicherzelle 14000 und druckt ihn auf den Bildschirm.

In der ersten Zeile wird der Inhalt der Variablen A zugeordnet und steht unter diesem Namen im Speicher für spätere Verwendung. Da der Inhalt aber trotz PEEKen in der Zelle 14000 stehenbleibt und von dort immer wieder herausgeholt werden kann, bietet sich die Kurzform der zweiten Zeile an, die direkt das PEEK-Ergebnis ausdruckt.

Basic-Befehl Nr. 28 PEEK

- er wird in der folgenden Weise geschrieben:
  PEEK (Adresse)
- die »Adresse« muß immer in Klammern stehen
- der Befehl liest den Inhalt der durch die »Adresse« angegebenen Speicherzelle
- der erlaubte Bereich von »Adresse« reicht von 0 bis 65635. Wird er überschritten, meldet dies der Computer mit ILLEGAL QUANTITY ERROR

Wir haben gerade vorhin mit dem POKE-Befehl die Zahl 55 in die Speicherzelle 14000 hineingeschrieben und sie danach mit PEEK wieder ausgelesen.

Nun, das beweist natürlich noch gar nichts. Schauen wir also mal im Speicher rund um die Zelle 14000 nach, was da so drin steht. Mit den Programmzeilen:

```
10 FOR I=13900 TO 14200
20 PRINT I,PEEK(I)
30 NEXT I
```

schauen wir uns den Speicherbereich von 13900 bis 14200 an. Es wird durch Zeile 20 der jeweilige Wert von I und daneben die Zahl, die in der Adresse I gespeichert ist, mit nur einem PRINT-Befehl ausgedruckt. Das Komma zwischen den beiden bewirkt einen Abstand von einer Viertelzeilenlänge.

Nach RUN sehen wir, daß in allen Speicherzellen entweder eine 0 oder 255 steht, mit Ausnahme der Zelle 14000, da steht brav unsere 55.

Übrigens hoffe ich, Sie wissen, daß mit der CTRL-Taste links oben der Ablauf des Programms gebremst werden kann – zum besseren Überblick, wenn die Speicherzelle 14000 vorbeisaust. In diesem Teil des Speichers steht also praktisch gar nichts. Das ist auch kein Wunder, denn wir tummeln uns ja im oberen Teil des RAM-Speichers, der uns für Basic-Programme zur Verfügung steht. Unser Mini-Programm von 3 Zeilen, welches ab der Speicherzelle 2048 gespeichert wird, reicht da natürlich bei weitem nicht hin. Wenn Sie Zeile 10 so abändern, daß der Ausdruck ab dem Speicheranfang 2048 beginnt:

```
10 FOR I=2048 TO 2200
```

dann sehen wir in der Tat ein Durcheinander von Zahlen, die bis hin zur Adresse 2095 reichen. Das ist – in einem besonderen Code geschrieben – unser Programm. Danach kommen wieder die Leerserien mit 0 und 255.

Schauen wir spaßeshalber noch in den obersten Teil des Speichers, wo laut Speicherbild die mysteriösen Register für Ein- und Ausgabe liegen. Ich wähle Speicherzelle 53265 als Versuchskaninchen.

```
PRINT PEEK (53265)
```

Das ergibt eine 155.

Jetzt machen wir ein Experiment – das, wie gesagt, unerwünschte Folgen haben kann. Wir ändern mutwillig den Inhalt dieser Speicherzelle mit:

```
POKE 53265,16
```

Und siehe da, nach Drücken der RETURN-Taste verschiebt sich der obere Rand des Bildschirms und schneidet alle Zeichen in der ersten Zeile ab. Sie sind zwar noch da, aber nicht sichtbar.

Der Originalzustand läßt sich mit dem oben ermittelten »Normalwert« wiederherstellen, indem Sie eintippen:

```
POKE 53265,155
```

Dieser Versuch ist gutgegangen. Aber wenn Sie statt der 15 oder der 155 die Zahl 33 in die Zelle POKEn:

```
POKE 53265,33
```

dann geht es schief. Der Bildschirm wird leer und durch nichts läßt er sich wiederbeleben – der Computer ist »abgestürzt«!

Ein Mittel bleibt uns doch, nämlich die <STOP RESTORE>-Tastenkombination.

FAZIT

1. Der POKE-Befehl ist nützlich und gefährlich zugleich.
2. Wird eine bestimmte Zahl in eine Speicherzelle gePOKEt, mit der der Computer seine eigenen Abläufe steuert (Bereiche 0 bis 2047, 40960 bis 49152 und 53248 bis 65535), kann dadurch der Ablauf beeinflußt werden. Es kann aber auch zum »Absturz« des Computers kommen.
3. Mit »Absturz« wird beim Computer der Zustand bezeichnet, in dem kein Programm mehr läuft und der Computer auf keine normalen Steuertasten mehr reagiert.
4. Ein abgestürzter Computer kann nur durch Aus- und Wiedereinschalten wieder in Gang gesetzt werden. Nach dem Aus- und Einschalten befindet sich der Com-

puter im Anfangszustand, das heißt, ein Programm, das vorher im Arbeitsspeicher war, ist verloren.
5. Es ist empfehlenswert, bevor ein POKE-Befehl ausgeführt wird, sei es im Direkt-Modus oder sei es innerhalb eines Programms, ein im Arbeitsspeicher befindliches Programm zuerst auf Band oder Diskette abzuspeichern – für alle Fälle !

21.3. Wir POKEn noch ein Weilchen

Es gibt keine Computerzeitschrift, die nicht unter der Rubrik Tips und Tricks alle möglichen POKE-Adressen angibt, mit denen sich verblüffende Effekte erzielen lassen. Auch ich gebe Ihnen ein paar Hinweise.

```
POKE 199,1:PRINT "ABCDE"
```

druckt alle Zeichen dieser Programmzeile revers (invertiert).

```
POKE 650,64
```

schaltet die Wiederholfunktion aller Tasten aus

```
POKE 650,0
```

nur die Leer-, INST/DEL- und alle Cursor-Tasten wiederholen, solange sie gedrückt werden

```
POKE 650,128
```

alle Tasten haben Wiederholfunktion (Normalzustand)

```
POKE 53281,4
```

schaltet die Farbe des Bildschirm-Hintergrundes auf Violett

```
POKE 53280,5
```

schaltet die Farbe des Bildschirm-Rahmens auf Grün.

Die beiden letzten POKE-Adressen 53281 und 53280 wollen wir uns näher anschauen.

Der Zahlenwert in der Speicherzelle 53281 bestimmt also die Hintergrundfarbe, während in Speicherzelle 53280 die Rahmenfarbe festgelegt ist. Da nach einem POKE-Befehl Zahlen von 0 bis 255 zugelassen sind, ist es sicher ganz interessant, welche Zahl welche Farbe hervorruft. Ein kleines Programm gibt uns darüber Auskunft:

```
10 FOR I=0 TO 255
20 POKE 53281,I
30 PRINT I
40 GET A$:IF A$="" THEN 40
50 NEXT I
```

Zwischen den Zeilen 10 und 50 wird in einer Schleife die Variable I von 0 bis 255 hochgezählt.

Der jeweilige Wert von I wird in Zeile 20 in die Speicherzelle 53281 gePOKEt und ändert dadurch die Hintergrundfarbe.

Um die Zugehörigkeit der Farben zu den Zahlen zu sehen, wird in Zeile 30 der jeweilige Wert von I ausgedruckt.

Zeile 40 dient dazu, die Schleife schrittweise weiterzuschalten. Der GET-Befehl in dieser Zeile springt so lange auf seine eigene Zeilennummer zurück, bis irgendeine beliebige Taste gedrückt wird. Erst dann kommt der NEXT-Befehl in Zeile 50 zum Zuge.

Mit diesem kleinen Programm werden pro Tastendruck alle Farben durchgeleiert, wobei die jeweils unterste ausgedruckte Zahl dem Farbwert entspricht.

Was wir vorher mit der Speicherzelle 53281 für die Hintergrundfarbe gemacht haben, können wir ebenso mit der Zelle 53280 für die Umrandung machen. Diese beiden Adressen gehören zu den »Registern« des VIC-Bausteins, der für Töne und Graphik zuständig ist.

FAZIT
1. Ein REGISTER ist eine Speicherzelle im Mikroprozessor (CPU) oder in einem anderen elektronischen Baustein. Im C64 besteht ein Register aus 8 Bit, das ist 1 Byte.
2. In einem Register werden Daten gespeichert, welche den Ablauf von arithmetischen, logischen oder von Steueroperationen festlegen.
3. Der Inhalt von Registern kann mit PEEK ausgelesen und, was viel wichtiger ist, mit POKE verändert werden.

Jetzt kennen wir also die beiden Farbregister und wissen, wie wir die Farben des Bildschirms unseren Wünschen anpassen können.

LEKTION 22: Der Bildschirmspeicher und Farbspeicher

Ich möchte Sie gern noch ein bißchen länger mit dem Speicher beschäftigen. Schauen Sie sich bitte nochmal das Speicherbild an. Da sehen wir ab Adresse 1024 das »Bildschirm-RAM«.

Um zu demonstrieren, was das ist, zeige ich Ihnen ein kleines Experiment:
- Löschen Sie den Bildschirm mit der CLEAR-Taste
- Fahren Sie mit dem Cursor in die untere Hälfte des Bildschirms.
- Geben Sie direkt ein:

```
POKE 1024,1
```

Ganz links oben auf dem Bildschirm steht plötzlich ein A. Wenn Sie aus der 1 eine 2 machen und den Befehl wiederholen, verwandelt sich das A in ein B.

Und noch ein Versuch: ändern Sie 1024,2 in 1025,3 um und geben es ein. Jetzt steht neben dem B ein C.

Die Zahl 1 erzeugt also ein A, 2 ein B und 3 ein C. Dann müßte eigentlich die 26 ein Z hervorrufen – was sie auch tut. Wir haben also einen neuen Code für die Zeichen auf dem Bildschirm gefunden.

Ein ähnlicher Zusammenhang deutet sich an bei den Adressen: 1024 setzt den Buchstaben an den ersten Platz des Bildschirmes, 1025 an den zweiten. Der Bildschirm hat 40 Stellen pro Zeile und das für 25 Zeilen. Das macht insgesamt 1000 Plätze auf dem Bildschirm.

Daher müßte die Adresse 2023 einen Buchstaben ganz rechts unten plazieren.

```
POKE 2023,26
```

setzt ein Z genau dorthin, wie vorhergesagt.

Ich will Sie nicht länger plagen und das alles zusammenfassen:

FAZIT
1. Der Elektronenstrahl, der mit großer Geschwindigkeit über den Bildschirm des Fernsehers oder des Monitors flitzt und dort Bilder oder Text hinmalt, hat kein Gedächtnis. Deswegen muß der Computer alle Angaben, die der Elektronenstrahl braucht, in einem gesonderten Speicher festhalten, der deshalb »Bildschirmspeicher« heißt.
2. Im Bildschirmspeicher sind alle Zeichen gespeichert, die zum jeweiligen Zeitpunkt auf dem Bildschirm erscheinen.
3. Für den Bildschirmspeicher sind im RAM die Speicherzellen 1024 bis 2047 reseviert. Da auf dem Bildschirm des C64 genau 1000 Plätze vorhanden sind (40 Stellen mal 25 Zeilen), reicht der Bildschirmspeicher nur bis Adresse 2123. Die restlichen 24 Byte sind frei.
4. Alle Zeichen und Buchstaben stehen im Bildschirmspeicher mit einem speziellen Code, der im Bedienungshandbuch auf den Seiten 133 und 134 aufgelistet und auf Seite 132 erklärt ist. Der Bildschirmcode hat nichts mit dem ASCII-Code zu tun.

Durch direktes POKEn von Bildschirm-Codewerten in den Bildschirmspeicher lassen sich Effekte erzielen, die mit dem PRINT-Befehl nur sehr umständlich möglich wären. Ich will Ihnen das an einem Beispiel zeigen.

Ziel des Experiments soll es sein, eine bewegte farbige Umrandung des Bildschirms zu erzeugen, die als Programmteil in anderen Programmen verwendbar ist. Ich verwende dazu ein Beispiel aus dem Buch »VC 20 Spielebuch« von A. Dripke, das viele gute Ideen und Anleitungen enthält. Wie üblich gehe ich in Stufen vor.

```
→ 10 FOR I=0 TO 999
  20 POKE 1024+I,42
  30 NEXT
```

Diese drei Zeilen zählen vom Anfang des Bildschirmspeichers (1024) 1000 Plätze hoch (von 0 bis 999) und POKEn in jeden Platz einen Stern. Der Bildschirmcode des Sternes ist 42 (siehe oben erwähnte Tabelle).

Auffallend bei diesem eindrucksvollen Vorgang ist, daß der Cursor und die READY-Meldung nicht anschließend an den letzten Stern erscheint, sondern dort, wo der letzte Basic-Befehl – in unserem Fall das RUN – auf den Bildschirm geschrieben worden ist.

Im nächsten Schritt lassen wir den Stern nur auf der obersten und untersten Zeile laufen. Die Schleife zählt daher jetzt nur bis 39 (Zeilenlänge):

```
→ 10 FOR I=0 TO 39
  20 POKE 1024+I,42
  30 POKE 1024+960+I,42
  40 NEXT I
```

Neu ist hier die Zeile 30. Sie erhöht die Adresse um 24*40=960 Plätze und beginnt dadurch in der untersten Zeile.

Dieses Programm malt also eine Rahmenlinie oben und unten. Jetzt fehlt noch links und rechts.


```
→ 50 FOR K=0 TO 960 STEP 40
  60 POKE 1024+K,42
  70 POKE 1024+39+K,42
  80 NEXT K
  90 GOTO 10
```

Um von oben nach unten zu zählen, beginnen wir mit 0, gehen aber in 40er Schritten gleich an den Anfang der jeweils nächsten Zeile bis zum Anfang der letzten Zeile. Das gibt uns in Zeile 60 den linken Rand.

Für die andere Seite benutzt die Zeile 70 die gleiche Zählung, POKEt aber den Stern um 39 Plätze verschoben, also am rechten Rand. Ganz zum Schluß springen wir in Zeile 90 auf den Anfang zurück, um das lästige READY zu verhindern.

Jetzt wollen wir diese Umrandung bunt machen, und zwar nach jedem Umlauf in einer anderen Farbe.

Auch hier bietet der Computer eine Lösung an. Zwischen den Adressen 55296 und 56319 liegt das »Farb-RAM«.

Dieser Bereich im RAM ist der Zwilling zum Bildschirmspeicher, nur ist er für die Farben zuständig.

Wenn wir jetzt das wiederholen, was wir gleich nach der Überschrift »Bildschirmspeicher« gemacht haben und POKEn zusätzlich in die erste Zelle des Farbspeichers 2048 die Zahl 100, dann ändern wir die Farbe des Zeichens, also:

```
→ - Bildschirm löschen
  - Cursor in die untere Hälfte
  - POKE 1024,1
  - POKE 55296,7
```

Das A erscheint in Gelb.

Insgesamt haben wir 16 Farben zur Verfügung. Sie entsprechen den folgenden Werten:

0 = schwarz	8 = orange
1 = weiß	9 = braun
2 = rot	10 = hellrot
3 = lila	11 = dunkelgrau
4 = purpur	12 = mittelgrau
5 = grün	13 = hellgrün
6 = blau	14 = hellblau
7 = gelb	15 = hellgrau

FAZIT

1. Im Farbspeicher sind alle Farben gespeichert, in welchem die Zeichen auf dem Bildschirm erscheinen.
2. Für den Farbspeicher sind im RAM die Speicherzellen 55296 bis 56319 reserviert.
   Da auf dem Bildschirm des C64 genau 1000 Plätze vorhanden sind (40 Stellen mal 25 Zeilen), reicht der Farbspeicher nur bis Adresse 55295. Die restlichen 24 Bytes sind frei.
3. Als Codewerte für die Farben gelten die Zahlen von 0 bis 15.

Wir wollten aber eigentlich das Umrahmungs-Programm mit Farben versehen. Dies erfolgt mit POKE in den Farbspeicher.

```
5 FOR F=0 TO 15
10  FOR I=0 TO 39
20   POKE 1024+I,42:POKE 55296+I,F
30   POKE 1024+960+I,42:POKE 55296+960+I,F
40  NEXT I
50  FOR K=0 TO 960 STEP 40
60   POKE 1024+K,42:POKE 55296+K,F
70   POKE 1024+39+K,42:POKE 55296+39+K,F
80  NEXT K
90 NEXT F
```

Listing 6. Farbige Umrandung

Ich habe folgendes gemacht:

Die Zeilen 5 und 90 bilden die übergeordnete Schleife, innerhalb der wir die Farbe F in den Farbspeicher POKEn.

Um sicherzustellen, daß die Farbe F in diejenigen Speicherzellen des Farbspeichers kommt, die denen des Bildschirmspeichers entsprechen, habe ich an jeden Bildschirm-POKE-Befehl ein POKE 55296 angehängt mit denselben Argumenten. Nur hinter dem Komma steht ein F für die Farbe und nicht die 42 für den Stern.

Sie sehen, die Arbeit des Ausrechnens der Adresse braucht man nur einmal zu machen.

Jetzt habe ich noch eine Variante vor:

Die Umrandung soll nicht oben und unten beziehungsweise links und rechts gleichzeitig laufen, sondern immer im Kreis.

Bisher hatten wir diese Situation:

Zeile 20: oben, von links nach rechts
Zeile 30: unten, von links nach rechts
Zeile 60: links, von oben nach unten
Zeile 70: rechts, von oben nach unten.

Wir müssen das so abändern:

Zeile 20: oben, bleibt
Zeile 70: rechts, bleibt
Zeile 30: unten, von rechts nach links
Zeile 60: links, von unten nach oben

Die Reihenfolge ändert sich also, und die Zeilen 30 und 60 laufen in der entgegengesetzten Richtung. Um das zu erreichen, müssen wir leider für jede POKE-Zeile eine eigene Schleife bauen. In den Zeilen 30 und 60 wird rückwärts gezählt, mit negativem STEP.

In der neuen Reihenfolge sieht das so aus (Vorsicht, neue Zeilennummern!):

```
5 FOR F=0 TO 15
10  FOR I=0 TO 39
20   POKE 1024+I,42:POKE 55296+I,F
30  NEXT I
40  FOR K=0 TO 960 STEP 40
50   POKE 1024+39+K,42:POKE 55296+39+K,F
60  NEXT K
70  FOR I=39 TO 0 STEP -1
80   POKE 1024+960+I,42:POKE 55296+960+I,F
90  NEXT I
100  FOR K=960 TO 0 STEP -40
110   POKE 11024K,42:POKE 55296+K,F
120  NEXT K
135 NEXT F
140 GOTO 5
```

Listing 7. Umlaufende Umrandung

LEKTION 23: Die Uhr des Computers

Der C64 hat eine innere Uhr eingebaut, deren Stand abgefragt, ausgedruckt und somit zu Messungen und zur Programmsteuerung eingesetzt werden kann. Diese Uhr startet beim Einschalten mit dem Stand 0 und läuft, bis sie durch einen entsprechenden Befehl auf Null oder auf irgendeinen anderen Wert gesetzt wird. Der aktuelle Stand dieser Uhr kann mit der dafür fest reservierten Variablen

TI

abgefragt werden. Mit der ewigen Schleife:

```
10 PRINT TI:GOTO 10
```

drucken wir ein laufendes Band von sich schnell ändernden Zahlen auf den Bildschirm.

Diese Zahl durch 60 geteilt gibt uns die Zeit seit dem Loslaufen (Einschalten) in Sekunden an, durch 3600 geteilt in Minuten und durch 216000 geteilt in Stunden.

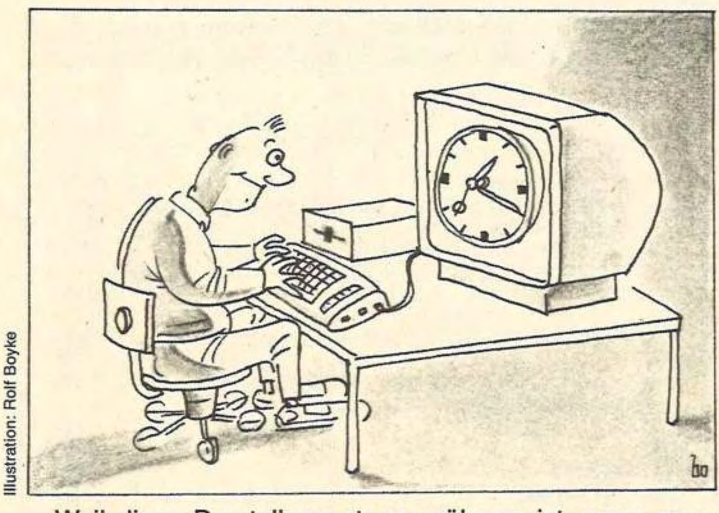


Illustration: Rolf Boyke

Weil diese Darstellung etwas mühsam ist, wenn man eine echte Zeitangabe braucht, besitzt Basic noch eine andere reservierte Variable

TI$

welche die Zeit in einer sechsstelligen Zahl ausdruckt. Dabei bedeutet 124533 12 Stunden, 45 Minuten und 33 Sekunden. Dies testen wir mit

```
10 PRINT TI$:GOTO 10
```

Dieses Zahlenband verändert sich jetzt im Rhythmus von Sekunden.

TI$ ist auch die Variable, mit dem die Uhr auf einen beliebigen Wert – auch auf Null – gesetzt wird:

```
TI$="000000"
```

setzt die Uhr auf Null.

```
TI$="221500"
```

setzt die Uhr auf 22 Uhr 15.

In dem Augenblick, wenn Sie nach diesen Befehlen die RETURN-Taste drücken, läuft die Uhr mit diesem neuen Anfangswert los.

Basic-Variable TI, TI$

- Beim Einschalten des Computers läuft ein interner Zähler los, der 60 mal in jeder Sekunde um 1 erhöht wird. Wenn der Zähler den Stand 5184000 erreicht hat – das entspricht einer Laufzeit von 24 Stunden, wird er auf 0 zurückgesetzt.
- Mit TI kann der Stand des Zählers abgefragt werden.
- Die Variable TI$ wandelt den Zahlenwert des Zählers in eine sechsstellige Zahl um, deren erste beiden Stellen die Stunden der Uhrzeit, die mittleren beiden die Minuten und die letzten beiden die Sekunden angeben.
- Mit der Anweisung TI$= "aabbcc" wird die interne Uhr auf aa Uhr bb Minuten cc Sekunden gestellt.

Mit einer Simulation einer Stoppuhr wollen wir die bei den Variablen TI und TI$ in einem Programm anwenden. Mit der Stoppuhr wollen wir den Vergleich der Laufzeiten der Zählschleife mit IF-THEN und der FOR-TO-NEXT-Schleife aus Lektion 8 wiederholen.

23.1. Stoppuhr

Unsere Stoppuhr läuft in dem Moment los, wo sie auf Null gesetzt wird:

```
10 TI="000000"
```

Sie stoppt mit der letzten Zeile der Messung:

```
90 PRINT TI:END
```

Um die Meßzeit in Sekunden zu erhalten, wenden wir noch obige Regel an, teilen durch 60 und drucken aus, was wir sehen werden. Zeile 90 wird verbessert zu:

```
90 PRINT TI/60 "SEKUNDEN"
```

Zwischen 10 und 90 plazieren wir unser Testprogramm. Ich schlage vor, genau wie in Lektion 8 den Bildschirm mit 375 A zu füllen.

Testprogramm »Stoppuhr«

ZÄHLER+GOTO	FOR-TO-NEXT
10 TI$="000000"	110 TI$="000000"
20 PRINT CHR$(147)	120 PRINT CHR$(147)
	130 FOR X=0 TO 374
40 PRINT "A";	140 PRINT "A";
50 IF X=374 THEN 80	
60 X=X+1	
70 GOTO 40	170 NEXT
80 PRINT	180 PRINT
90 PRINT TI/60"SEK"	190 PRINT TI/60"SEK"
99 END	199 END

Die ungleichen Lücken zwischen den Zeilen, sowohl in der Numerierung als auch in der Schreibweise, dienen nur der Lesbarkeit, haben aber auf den Ablauf keinen Einfluß.

Die 80er Zeilen rücken den Ausdruck des Ergebnisses nach unten.

Ergebnis:
ZÄHLER+GOTO-Schleife: 3.9333 SEK.
FOR-NEXT-Schleife: 0.9666 SEK.

Die FOR-NEXT-Schleife ist strahlender Sieger !!

Diese Technik wollen wir verwenden, um eine Zeitbegrenzung so in ein Programm einzubauen, daß es nach einer gewissen Laufzeit abgebrochen, beziehungsweise beendet wird.

23.2. Zeitabhängige Programmunterbrechung

Wenn das Programm im Prinzip aus einer oder mehreren

Schleifen besteht, ist die Lösung einfach. Ist es ein gerade verlaufendes Programm, das an irgendeiner Stelle unterbrochen werden soll, wird die Sache schon schwieriger, weil die Abfrage der Zeit eigentlich dauernd erfolgen muß.

Wir nehmen hier den leichteren Fall an die Reihe. Sie kennen doch sicher das alte Spiel »Stadt, Land, Fluß«, bei dem innerhalb einer festgesetzten Zeit aus jedem Sachgebiet ein Beispiel mit denselben Anfangsbuchstaben aufzuschreiben ist. Eine Abwandlung dieses Spiels wähle ich als Anwendung einer Zeitsteuerung, die ich – wie immer – schrittweise mit Ihnen entwickeln möchte.

1. Initialisierung

Mit diesem Fremdwort bezeichnen wir das Herstellen des Anfangszustandes eines Programms. In unserem Fall soll festgelegt werden:
- ein Sachgebiet (S$)
- die Zeitdauer (Z)
- der Anfangsbuchstabe (B$)

Sachgebiet und Zeitdauer werden vom Spieler per INPUT eingegeben, der Buchstabe (von A bis Z) wird mit einem Zufallsgenerator erzeugt.

```
100 PRINT CHR$(147)
110 INPUT "WAEHLE EIN SACHGEBIET";S$
120 INPUT "STELLE DIE UHR AUF";Z
130 B$=CHR$(INT(RND(0)*26+65))
```

Ich glaube, zu den Zeilen 100 bis 120 ist nichts weiter zu sagen.

Zeile 130 soll eine Zufallszahl erzeugen, deren ASCII-Code laut Tabelle auf Seite 135 des Bedienungshandbuches zwischen 65 (A) und 90 (Z) liegt. Das Kochrezept dazu habe ich in Lektion 10 erklärt. Seine Formel lautet wie folgt:

Ganze Zahlen innerhalb des Zahlenbereichs von minimal Y bis maximal (Y+X-1) werden erzeugt durch:

INT(RND(0)*X+Y)

Zeile 130 wendet diese Formel an. Y liegt mit 65 fest, X errechnet sich aus (Y+X-1)=95 und ergibt 26. Bei Zeile 130 bitte auf die Anzahl der Klammern aufpassen!

Zeile 140 faßt die Initialbedingungen zusammen. Sie ist ein Beispiel für das Einfügen von Wörtern in einen vorgegebenen Text. Wichtig sind dabei die Leerzeichen im Text vor und nach den Stringvariablen S$ und B$. Versuchen Sie es ohne Leerzeichen, und Sie werden das unlesbare Resultat sehen.

Zeilen 230, 240 und 250 geben den Startschuß.

```
→ 140 PRINT"NENNE "S$", DIE MIT "B$" ANFANGEN"
  230 PRINT"START MIT IRGENDEINER TASTE":
  240 GET X$:IF X$=""THEN 240
  250 TI$="000000"
```

Jetzt läuft eine Zeitschleife los, die wie im letzten »Weckprogramm« einen vorgegeben Zeitwert Z mit dem Stand der eingebauten Uhr TI$ vergleicht.

```
→ 270      (Wörter eingeben)
  340 IF VAL(TI$) <> THEN 270
```

Innerhalb der Zeilen 270 und 340 soll Folgendes passieren:
- Wörter W$ eingeben
- Anfangsbuchstaben B$ überprüfen
- Wörter ausdrucken
- richtige Wörter zählen (R)

Zum Eingeben von Wörtern steht uns INPUT und GET zur Verfügung.

INPUT erlaubt die Eingabe von kompletten Wörtern, indem es auf einzelne Zeichen wartet – macht aber damit die Wirkung der Zeitmeß-Schleife zunichte.

GET wartet nicht und ermöglicht der Zeitschleife, in den Ablauf einzugreifen – nimmt aber nur einzelne Zeichen entgegen. Die Wirkung der Zeitschleife ist mir wichtiger, zumal man aus einzelnen Zeichen auch Wörter zusammensetzen kann! Wir nehmen also GET (Zeile 270).

Das Zusammensetzen eines Wortes besorgt uns Zeile 280, indem die einzelnen Buchstaben E$ zu einem Wort W$ einfach dazuaddiert werden:

```
→ 270 GET S$
  280 W$=W$+S$
```

Mit den Zeilen 270, 280 und 340 haben wir eine Schleife, die solange Wörter zusammensetzen würde, bis der Zeitvergleich alles abbricht. Versuchen Sie es mit RUN 270!

Ein Wort wird dadurch abgeschlossen – wie bei INPUT – daß die RETURN-Taste gedrückt wird. Dann allerdings soll das Wort ausgedruckt und als Resultat (R) gezählt werden – wenn es richtig ist. Die primitivste Prüfung ist die des Anfangsbuchstabens.

```
→ 290 IF E$ <> CHR$(13) THEN 340
```

Zeile 290 prüft auf Drücken der RETURN-Taste, die den ASCII-Code 13 hat. Das heißt, sie prüft, ob die Taste nicht gedrückt ist. Dann rückt das Programm auf Zeile 340 weiter, mißt die Zeit, springt auf Zeile 270 zurück und schaut nach, ob ein neuer Buchstabe E$ eingegeben worden ist.

Ist <RETURN> aber gedrückt, geht das Programm mit der Zeile 300 weiter. Diese schneidet sich mit dem LEFT$-Befehl den ersten Buchstaben des Wortes W$ ab und vergleicht ihn mit dem vorgegebenen Anfangsbuchstaben B$.

```
→ 300 IF LEFT$(W$,1) <> B$ THEN 330
  310 PRINT W$
  320 R=R+1
  330 W$=""
```

Ist die Prüfung auf »ungleich« nicht erfüllt, das heißt, sind beide Buchstaben gleich, dann druckt Zeile 310 das Wort W$ aus und zählt es als richtiges Resultat R zu eventuell schon vorhandenen richtigen Resultaten dazu. Danach wird das eingegebene Wort W$ in Zeile 330 wieder gelöscht.

Ergibt der Vergleich der beiden Buchstaben in Zeile 300 aber, daß sie ungleich sind, dann überspringt Zeile 300 diesen ganzen Teil, löscht in Zeile 330 das bisher eingegebene (aber falsche) Wort und macht in Zeile 340 mit der Schleife weiter.

Nach Ablauf der Schleife, oder besser gesagt, nach Abbruch durch die Uhr, läßt Zeile 430/440 den Bildschirm-

```
100 PRINT CHR$(147)                          <129>
110 INPUT"WAEHLE EIN SACHGEBIET";S$          <235>
120 INPUT"STELLE DIE UHR AUF";Z              <082>
130 B$=CHR$(INT(RND(0)*26+65))               <006>
140 PRINT"NENNE "S$", DIE MIT " B$ " ANFAN
    GEN                                      <010>
200 :                                        <176>
210 REM**** ZEITSCHLEIFE & EINGABE ****      <029>
220 :                                        <196>
230 PRINT"START MIT IRGENDEINER TASTE"       <158>
240 GET X$:IF X$=""THEN 240                  <214>
250 TI$="000000"                             <229>
260 :                                        <238>
270 GET E$                                   <002>
280 W$=W$+E$                                 <152>
290 IF E$<>CHR$(13) THEN 340                 <039>
300 IF LEFT$(W$,1)<>B$ THEN 330              <200>
310 PRINT W$;                                <001>
320 R=R+1                                    <230>
330 W$=""                                    <193>
340 IF VAL(TI$)<>Z THEN 270                  <239>
400 :                                        <122>
410 REM**** STOP & ERGEBNIS ******           <091>
420 :                                        <142>
430 FOR F=0 TO 14                            <014>
440 POKE 53280,F:NEXT F                      <120>
450 PRINT R                                  <188>
```


Listing 8. Die eingebaute Uhr

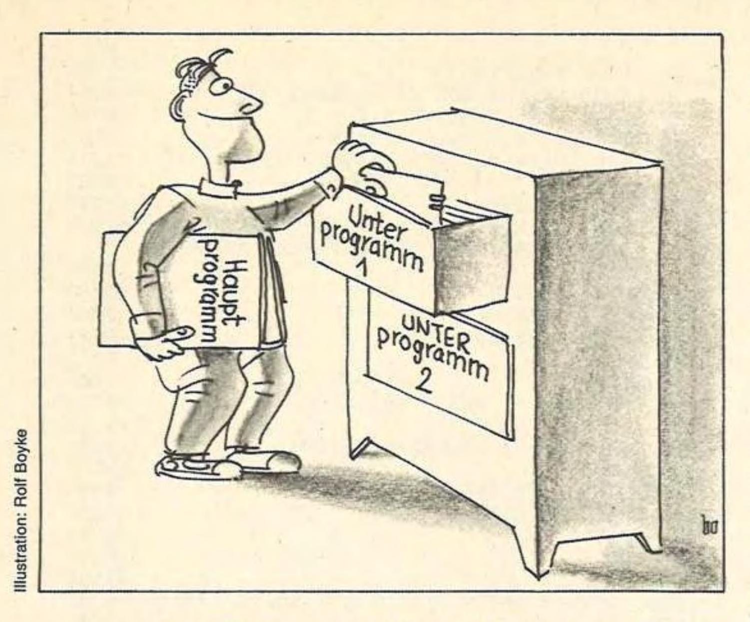


Illustration: Rolf Boyke

rahmen bunt aufflimmern und Zeile 450 druckt das Resultat R, nämlich die Anzahl der eingegebenen Wörter aus:

```
430 FOR F=0 TO 14
440 POKE 53280,F:NEXT F
450 PRINT R
```

In Listing 8 ist das bisherige Programm vollständig dargestellt. Ich habe es dabei mit ein paar REM-Zeilen in funktionelle Blöcke unterteilt.

LEKTION 24: Die Technik der Unterprogramme

Das Spiel »Stadt, Land, Fluß« wird immer mit mehreren Personen gespielt. Obwohl mir klar ist, daß ein derartiges Vorhaben mit dem Computer nicht ganz einfach zu organisieren ist – kein Spieler soll die Wörter des vorhergehenden Spielers sehen können – lasse ich dieses Detail außer acht und zeige Ihnen, wie man so einen mehrfachen Ablauf desselben Programms programmiert.

Wir brauchen dazu folgende Schritte:
- Initialisierung, wie vorher
- Frage nach der Zahl der Mitspieler
- Schleife für entsprechend viele Durchgänge
- Einzeldurchgänge pro Spieler
- Speicherung der Einzelergebnisse
- Ausdruck des Gesamtergebnisses

Die Initialisierung mit den Zeilen 100 bis 130 bleibt also gleich.

Jetzt stehen wir vor dem Problem, den eigentlichen Spielteil von Zeile 300 bis Zeile 340 so oft laufen zu lassen, wie Mitspieler vorhanden sind. Völlig unsinnig wäre es, diesen Programmteil zu vervielfachen, da ja die Anzahl der Mitspieler variabel sein soll.

Also müssen wir den Spielteil, der durch die Zeilen 200 bis 340 gebildet wird, bei jedem Spieldurchgang neu »anspringen«. Dazu legen wir ihn zuerst einmal an das Ende des Programms. Das geht am schnellsten dadurch, daß Sie vor jede Zeilennummer eine 1 einfügen, wodurch wir einen Programmblock von 1200 bis 1340 erhalten, die Abstands- und REM-Zeilen mit eingeschlossen.

Vorsicht! Wir haben innerhalb dieser Zeilen zwei Sprungadressen, die wir ebenfalls ändern müssen: in Zeile 290 und in Zeile 300.

Die »alten« Zeilen 200 bis 340 müssen wir durch Eintippen ohne Inhalt der Reihe nach löschen.

Wenn Sie das Programm LISTen, dann sitzt in der Tat der eigentliche Spielteil jetzt am Ende des Programms, und Zeilen 200 bis 340 sind verschwunden

Dasselbe machen wir mit dem anderen Programm-Teilchen »STOP & ERGEBNIS«, das wir ebenfalls öfters verwenden wollen. Wir schieben es mit der oben genannten Methode auf 1400 bis 1450 und löschen Zeilen 400 bis 450.

Jetzt haben wir Platz, um die Frage nach der Anzahl der Mitspieler einzubauen. Ich schiebe die Frage noch vor die Spielanweisung in Zeile 140, die ihrerseits in Zeile 220 rutscht.

```
→ 140 INPUT "WIEVIELE MITSPIELER";MS
  150 SP=1
```

MS ist also die festgelegte Mitspielerzahl, die während eines Spiels konstant bleibt. Da wir aber mit Spieler 1 anfangen und mit Spieler M$ aufhören, brauchen wir noch eine Spieler-Zählvariable, die pro Durchlauf hochgezählt wird. Ich nenne sie »SP« und setze sie in Zeile 150 auf 1.

Als nächstes folgt die große Schleife für die Spieldurchgänge pro Mitspieler. Ihre Zahl ist natürlich von MS abhängig:

```
→ 70 SP=SP+1
  280 IF SP<> MS+1 THEN 210
```

Die Schleife zwischen den Zeilen 210 und 280 läuft solange, bis SP, welches am Anfang 1 ist und nach jedem Durchgang in Zeile 270 um 1 weitergezählt wird, die volle Mispielerzahl MS erreicht hat (Prüfung in Zeile 210 auf MS+1).

In die Schleife lege ich jetzt die Anweisung der alten Zeile 140 mit der Spielanleitung und welcher Mitspieler an der Reihe ist:

```
→ 210 PRINT"MITSPIELER "SP" IST AN DER REIHE"
  220 PRINT"NENNE " S$ ", DIE MIT " B$
      "ANFANGEN"
```

Nach Zeile 220 muß jetzt der Sprung auf den Spielteil erfolgen, den wir ans Ende des Programms zwischen den Zeilen 1230 und 1340 geschoben haben.

Bislang sind wir auf solche Stellen mit GOTO 1230 hin- und mit GOTO xxx wieder zurückgesprungen. Basic kennt aber einen Befehl, der das viel eleganter macht, den sogenannten »Unterprogrammsprung«. Er heißt

GOSUB-RETURN

wobei »Sub« vom englischen »Subroutine« kommt.

Der Befehl GOSUB steht anstelle des GOTO, RETURN kommt ohne weitere Angaben an den Schluß des anzuspringenden Programmteils, also dorthin, wo wir das zweite GOTO hingesetzt hätten. In unserem Fall sieht das so aus:

```
→ 230 GOSUB 1230
  1350 RETURN
```

Wichtig ist noch zu erwähnen, daß der Programmteil zwischen den Zeilen 1230 und 1350 Unterprogramm heißt.

Wichtig ist außerdem, daß der RETURN-Befehl dann automatisch auf die Zeile nach dem GOSUB springt.

Sinn und Zweck der Unterprogramm-Technik ist, Programmteile zu bilden, die man nicht nur innerhalb eines Programms mehrfach verwenden kann, sondern die auch in sich beliebig verändert werden können, ohne das Hauptprogramm zu stören. Alles was bekannt beziehungsweise konstant bleiben muß, ist die Anfangs-Zeilennummer des Unterprogramms.

Ein weiterer Vorteil eines Unterprogramms ist, daß es einzeln auf Band oder Diskette in einer »Unterprogramm-Bibliothek« gespeichert und in anderen Programmen eingesetzt werden kann, ohne es immer wieder neu programmieren zu müssen.

Wir erklären den Programmteil »STOP & ERGEBNIS« ab Zeile 1430 ebenfalls zum Unterprogramm und springen nach dem ersten Unterprogramm »ZEITSCHLEIFE & EIN-

GABE« mit einem zweiten GOSUB-Befehl dorthin. Voraussetzung ist, daß es mit RETURN abgeschlossen ist:

```
→ 240 GOSUB 1430
  1500 RETURN
```

Wenn Sie das Programm jetzt schon laufen lassen, werden Sie sehen, daß nur noch wenig fehlt. Zum einen werden die Ergebnisse der einzelnen Mitspieler aufaddiert, zum zweiten fehlt noch eine Rücksetzung vor jedem neuen Spieler, und als letztes brauchen wir noch ein Gesamtergebnis. Diese Dinge sind aber nichts Neues für uns.

```
→ 250 PRINT CHR$(147)
  260 R=0
```

Diese beiden Zeilen besorgen das Löschen und die Rücksetzung des Einzelresultats R. Dieses R müssen wir aber speichern, um es am Ende ausdrucken zu können. Das machen wir am Schluß des zweiten Unterprogramms, wo wir ja in Zeile 1450 das Einzelergebnis R ausdrucken.

Das Gesamtergebnis besteht also aus soviel Einzelergebnissen, wie Mitspieler vorhanden sind. Das schreit nach einem Feld!

Da sicher weniger als 11 Mitspieler vorzusehen sind, brauchen wir das Feld nicht zu dimensionieren. Wir teilen direkt einer Feldvariablen R(SP) die jeweiligen Einzelresultate zu, wobei SP ja von 1 bis MS hochgezählt wird. Dadurch werden alle anfallenden Werte von R gespeichert.

```
→ 1450 R(SP)=R
  1460 PRINT R(SP)
  1470 PRINT"ZUR FORTSETZUNG TASTE DRÜCKEN"
  1480 GET A$:IF A$="" THEN 1480
```

Im Schlußteil wird das Endergebnis ausgedruckt. Es steht, wie gesagt, in einem Feld R(0), R(1),...R(letzter Spieler), das wir mit einer FOR-NEXT-Schleife ausdrucken:

```
→ 430 FOR I=1 TO (SP-1)
  440 PRINT"SPIELER "I" HAT: "R(I)
  450 NEXT I
  460 END
```

Zeile 460 ist wichtig, da sie das Hauptprogramm von den Unterprogrammen trennt.

Dieses Programm hat uns also in die Technik der Unterprogramme eingeführt, außerdem haben wir durch eine Schleife mit Abfrage der inneren Uhr des C64 eine Zeitsteuerung des Programms erreicht, haben Farbänderungen eingesetzt und wir haben letztlich wieder einmal eine Feldvariable verwendet.

In Listing 9 ist das ganze Programm zusammengefaßt.

Basic-Befehl Nr. 28 und 29 GOSUB-RETURN

- der Befehl wird geschrieben:
  GOSUB Zeilennummer
- er erzeugt einen Sprung in ein Unterprogramm, welches mit der hinter dem GOSUB stehenden Zeilennummer anfängt
- das Unterprogramm muß mit RETURN in der letzten Zeile abgeschlossen werden
- RETURN springt zurück auf die Zeile, die direkt hinter dem GOSUB-Befehl steht
- Unterprogramme können auch geschachtelt werden
- Unterprogramme können sich selbst aufrufen, aber nur bis zu 25 mal. Darüber hinaus ist zuwenig Speicherplatz zur Zählung der Aufrufe vorhanden
- trifft ein Programm auf ein RETURN, ohne vorher ein GOSUB gesehen zu haben, reagiert der Computer mit Abbruch und Fehlermeldung »RETURN WITHOUT GOSUB«

In Listing 9 haben wir die beiden Unterprogramme an das Ende gelegt. Das war leicht zu erreichen durch einfache Umnumerierung der Zeilen. Wir mußten aber ans Ende des Hauptprogramms – noch vor den Unterprogrammen – einen END-Befehl einfügen, um ein unerwünschtes Weiterlaufen des Programms zu verhindern.

```
1ØØ PRINT CHR$(147)                           <129>
11Ø INPUT"WAEHLE EIN SACHGEBIET";S$           <235>
12Ø INPUT"STELLE DIE UHR AUF";Z               <Ø82>
13Ø B$=CHR$(INT(RND(Ø)*26+65))                <ØØ6>
14Ø INPUT"WIEVIELE MITSPIELER";MS             <197>
15Ø SP=1                                      <182>
2ØØ REM SCHLEIFE                              <124>
21Ø PRINT"SPIELER "SP" IST AN DER REIHE"      <118>
22Ø PRINT"NENNE "S$" DIE MIT "B$" ANFANGEN
    "                                         <Ø8Ø>
23Ø GOSUB 123Ø                                <Ø58>
24Ø GOSUB 143Ø                                <1ØØ>
25Ø PRINT CHR$(147)                           <Ø23>
26Ø R=Ø                                       <231>
27Ø SP=SP+1                                   <Ø14>
28Ø IF SP<>MS+1 THEN 2ØØ                      <Ø77>
4ØØ :                                         <122>
41Ø REM ***** ENDERGEBNIS *****               <Ø8Ø>
42Ø :                                         <142>
43Ø FOR I=1 TO (SP-1)                         <Ø74>
44Ø PRINT"SPIELER "I" HAT: "R(I)              <245>
45Ø NEXT I                                    <Ø24>
46Ø END                                       <2Ø8>
12ØØ :                                        <16Ø>
121Ø REM**** ZEITSCHLEIFE & EINGABE ****      <Ø13>
122Ø :                                        <18Ø>
123Ø PRINT"START MIT IRGENDEINER TASTE"       <142>
124Ø GET X$:IF X$=""THEN 124Ø                 <Ø87>
125Ø TI$="ØØØØØØ"                             <213>
126Ø REM SCHLEIFE                             <168>
127Ø GET E$                                   <24Ø>
128Ø W$=W$+E$                                 <136>
129Ø IF E$<>CHR$(13) THEN 134Ø                <2ØØ>
13ØØ IF LEFT$(W$,1)<>B$ THEN 133Ø             <125>
131Ø PRINT W$;                                <241>
132Ø R=R+1                                    <214>
133Ø W$=""                                    <177>
134Ø IF VAL(TI$)<>Z THEN 127Ø                 <Ø49>
135Ø RETURN                                   <138>
14ØØ :                                        <1Ø6>
141Ø REM**** STOP & ERGEBNIS ******           <Ø75>
142Ø :                                        <126>
143Ø FOR F=Ø TO 14                            <254>
144Ø POKE 5328Ø,F:NEXT F                      <1Ø4>
145Ø R(SP)=R                                  <114>
146Ø PRINT"ZUR FORTSETZUNG TASTE DRUECKEN"    <2Ø3>
147Ø GET A$:IF A$=""THEN 147Ø                 <253>
15ØØ RETURN                                   <Ø32>
```


Listing 9. Das fertige Stadt-Land-Fluß

Es gibt nun auch die Möglichkeit, Unterprogramme prinzipiell an den Anfang eines Programms zu legen. Das vermeidet den Weiterlauf und hat außerdem eine kürzere Laufzeit. Ich habe das in dem Kurs »So macht man Programme schneller« im Sonderheft 2/1986 auf Seite 49 gemessen und beschrieben.

Ein Programm sieht dann so aus:

```
GOTO "Hauptprogramm"

  Unterprogramm U1
  RETURN

  Unterprogramm U2
  RETURN

"Hauptprogramm"
GOSUB U1
GOSUB U2
```

Diese Anordnung ist sicher nur dann empfehlenswert, wenn es auf Geschwindigkeit ankommt, und wenn viele Unterprogramme verwendet werden.



Programme, die mit vielen Unterprogrammen arbeiten, verwenden oft eine eigene Technik, um »bedienungsfreundlich« zu sein. Mit diesem Begriff bezeichnet die Computerwelt Programme, deren Bedienung so ausgelegt ist, daß man von den Details des Programms oder vom Computer selbst praktisch nichts zu verstehen braucht. Ein wesentlicher Bestandteil davon sind die sogenannten »Menüs«.

Ich schreibe diesen Kurs auf meinem C64 mit einem bekannten Textprogramm, das sich beim Einschalten mit folgendem Menü meldet:

F1 Text bearbeiten
F3 Neuen Text eingeben
F5 Disk Inhalt
F7 Disk Befehle
F8 Ende

Je nachdem, welche Funktionstaste ich drücke, kann ich einen bereits angefangenen Text weiterbearbeiten oder mir anschauen, welche Programme sich auf der Diskette befinden oder spät in der Nacht mit <F8> Schluß machen!

Jede Überschrift bildet ein eigenes Unterprogramm, auf das durch Drücken der vorgegebenen Funktionstaste gesprungen wird.

Mit unseren heutigen Kenntnissen würden wir diese 5 Funktionstasten mit 5 IF-THEN-Befehlen abfragen. Größere Menü bräuchten dementsprechend mehr IF-THEN-Zeilen.

In Basic gibt es einen Befehl, mit dem dies sehr viel leichter programmiert werden kann. Er heißt schlicht und einfach

ON

und wird in Verbindung mit dem GOTO oder GOSUB-Befehl verwendet.

Ein ON-GOSUB-Befehl sieht so aus:

ON X GOSUB 100,150,200,250

Diese Zeile ist gleichbedeutend mit:

IF X=1 GOSUB 100
IF X=2 GOSUB 150
IF X=3 GOSUB 200
IF X=4 GOSUB 250

Die Variable X darf also nur die Werte 1, 2, 3, 4 etc. annehmen. Entsprechend springt der GOSUB-Befehl auf die erste, zweite, dritte oder vierte Zeilennummer.

Hinter dem ON-Befehl kann auch ein Formelausdruck stehen, nur gilt bezüglich seines Resultats das gleiche wie für die Variable. Es ist klar, daß für Entscheidungen zu Unterprogrammsprüngen nicht immer derartige »saubere« Zahlen zur Verfügung stehen. Damit aber die Umrechnung in die Werte 1, 2, 3 usw. nicht aufwendiger als mehrere IF-GOSUB-Befehle wird, werden oft sehr pfiffige Rechen- und Programmierverfahren angewendet. Sie sind es fast immer wert, gesammelt zu werden, wenn man sie in einem Programmlisting antrifft.

Basic-Befehl Nr. 30 ON-GOSUB
- der Befehl wird so geschrieben:
  ON Variable GOSUB mehrere Zeilennummern
  wobei die Zeilennummern durch Kommata getrennt sein müssen
- die Variablenwerte legen fest, auf welche Zeilennummer der GOSUB-Befehl springt und zwar:
  1.0 bis 1.9 — 1.Zeilennummer hinter dem GOSUB
  2.0 bis 2.9 — 2.Zeilennummer ...
  3.0 bis 3.9 — 3.Zeilennummer ...
  und so weiter
- ist die Variable kleiner als 1, springt der GOSUB-Befehl nicht auf eine der Zeilennummern, sondern auf die nach dem ON-GOSUB folgende Zeile
- ist die Anzahl der Variablenwerte größer als die Anzahl der Zeilennummern, verhält sich der ON-GOSUB-Befehl wie beim Variablenwert 0 oder kleiner als 1
- einen negativen Variablenwert quittiert der Computer mit ILLEGAL QUANTITY.
- hinter dem ON-Befehl kann auch eine Formel oder eine Funktion stehen, nur gilt für ihre Werte dasselbe wie für die Werte einer Variablen.

Der Befehl ON darf auch in Verbindung mit dem GOTO-Befehl verwendet werden.

Basic-Befehl Nr. 31 ON GOTO
Für ON GOTO gilt das gleiche wie für ON GOSUB, nur werden die Sprünge nach den Regeln des GOTO-Befehls ausgeführt

LEKTION 25: Positionierung des Cursors

Ich möchte Ihnen noch schnell die Programmierung des obigen Menü zeigen. Ein Menü beginnt immer mit seiner Darstellung auf dem Bildschirm. Das geht einfach über PRINT-Befehle, die nur wenige Zeilen auf dem Bildschirm verteilen müssen. Da es lästig ist, mit programmierten Cursor-Tasten zu operieren, gibt es noch zwei andere Cursorbefehle, die wir gleich anwenden wollen. Der eine heißt

TAB()

abgeleitet von »Tabulator«, der andere heißt

SPC()

was vom englischen Wort »Space«, das heißt »Abstand« kommt. In der Klammer hinter den Befehlen kann eine Zahl von 0 bis maximal 255 stehen.

Beide Befehle werden zusammen mit dem PRINT-Befehl verwendet. Beim TAB-Befehl rückt der Cursor ausgehend vom linken Rand derjenigen Zeile, auf der er sich gerade befindet, um die in der Klammer stehende Anzahl von Leerstellen weiter. Der SPC-Befehl tut dasselbe, aber vom Punkt aus, auf dem sich der Cursor gerade befindet.

```
10 PRINT CHR$(147)
20 PRINT SPC(90)"WAEHLEN SIE BITTE AUS"
```

CHR$(147) löscht bekanntlich nicht nur den Bildschirm, sondern setzt auch den Cursor in die linke obere Ecke. Von dort aus springt er durch SPC(90) 90 Plätze weiter, das sind 80 (also 2 Zeilen) und 10 Plätze. Sie können das leicht nach RUN auf dem Bildschirm abzählen.

```
30 PRINT TAB(202)"(1)   TEXT BEARBEITEN"
```

Nach Ausführung der Zeile 20 steht der Cursor in der 3. Zeile. 202 Plätze vom linken Rand der 3. Zeile aus gerechnet sind 5 Zeilen und 2 Plätze; also steht der Cursor in der 9. Zeile am 2. Platz, und die Klammer vor der 1 wird am 3. Platz gedruckt.

Ich will Ihnen gestehen, ich habe es auch nicht gerechnet, sondern einfach ausprobiert.

Die nächsten Zeilen verwenden in gleicher Weise den SPC-Befehl.

```
40 PRINT SPC(82)"(2)   NEUEN TEXT EINGEBEN"
50 PRINT SPC(82)"(3)   DISK INHALT"
60 PRINT SPC(82)"(4)   DISK BEFEHLE"
70 PRINT SPC(82)"(5)   ENDE"
```

Die Unterprogramme, die wir über das Menü auswählen wollen, lassen wir ab Zeile 500 beginnen. Ich deute sie natürlich nur an:

```
→ 500 PRINT CHR$(147)
  510 PRINT "TEXT BEARBEITEN"
  520 END
  530 :
  540 PRINT CHR$(147)
```

```
550 PRINT "NEUEN TEXT EINGEBEN"
560 END
570 :
580 PRINT CHR$(147)
590 PRINT"DISK INHALT"
600 END
610 :
620 PRINT CHR$(147)
630 PRINT"DISK BEFEHLE"
640 END
650 :
660 PRINT CHR$(147)
670 PRINT"ENDE"
680 END
```

Die Unterprogramme beginnen also der Reihe nach bei 500, 540, 580, 620 und 660. Dementsprechend lautet der ON-GOSUB-Befehl:

```
→ 110 ON A GOSUB 500, 540, 580, 620, 660
  120 PRINT"NULL"
  130 END
```

Zeile 120 ist die Ziel-Zeile im Fall, daß für A ein Wert kleiner als 1 eingegeben wird.

Die Eingabe mache ich gewöhnlich mit GET A.

In diesem Fall aber, wo ich Ihnen ermöglichen möchte, auch mit krummen Werten von A zu experimentieren, machen wir es mit INPUT.

```
→ 100 INPUT A
```

So einfach ist das. Bitte experimentieren Sie ein bißchen mit diesem Programm.

Basic-Befehle Nr. 32 und 33 TAB(A) SPC(A)
- hinter den beiden Befehlen steht eine Zahl von 0 bis 255. Sie muß in Klammern stehen
- TAB(I) und SPC(I) werden in Verbindung mit dem PRINT-Befehl verwendet
- mit PRINT TAB(I) beginnt der Ausdruck auf dem Bildschirm (oder auf dem Drucker) an dem durch I definierten Platz
  I=0 definiert den linken Rand der Zeile, auf welcher der Cursor sich gerade befindet, mit I=20 werden 20 Plätze vom linken Rand aus übersprungen
- mit PRINT SPC(I) beginnt der Ausdruck um genau soviele Plätze hinter der augenblicklichen Position des Cursors, wie durch I definiert werden
  I=0 bedeutet die augenblickliche Cursorposition, mit I=20 werden 20 Plätze übersprungen
- beide Befehle springen auf den angegebenen Platz, ohne die übersprungenen Plätze zu löschen

Bei dieser Gelegenheit bietet es sich an, einen weiteren Basic-Befehl zu erwähnen, der wie SPC und TAB mit der Position des Cursors auf dem Bildschirm zu tun hat, allerdings in anderer Weise. Er heißt

POS()

was von POSITION abgeleitet ist. Seinem Namen entsprechend liefert er die aktuelle Position des Cursors auf dem Bildschirm.

Die Sache mit dem Cursor ist nicht ganz so genau zu nehmen. Es gibt ja Möglichkeiten innerhalb eines Programms, wo der Cursor überhaupt nicht auftaucht, zum Beispiel bei einer Stringmanipulation. In diesen Fällen zeigt POS die Position hinter dem gerade bearbeiteten Zeichen an. Denken Sie bitte daran, daß die Positionen in einer Zeile von 0 bis 39 gezählt werden. Ein Beispiel macht das viel klarer:

```
PRINT"01234567" POS(0)
```

Dieser Direktbefehl druckt zuerst den String, bestehend aus den Ziffern 0 bis 7 aus und dann die Position hinter dem letzten Zeichen, und das ist die 8.

Übrigens: hinter POS steht eine Klammer. Die Variable – auch Argument genannt – hat keine Bedeutung, es muß nur irgendeine Variable oder ein beliebiger Wert in der Klammer stehen. Sie wird im Fachjargon »Dummy«-Argument genannt.

Leider hat der POS-Befehl eine Einschränkung. Er wirkt nur innerhalb einer »logischen« Zeile, nicht aber auf dem ganzen Bildschirm. Oder anders ausgedrückt, er wirkt nur waagrecht, nicht aber senkrecht, mit Ausnahme der Logischen Zeile. Was das ist, haben wir in Lektion 5 gelernt – es ist eine durch eine Zeilennummer gekennzeichnete Programmzeile, die maximal aus zwei vollen Bildschirmzeilen bestehen kann. Der POS-Befehl gibt als Resultat daher Zahlen von 0 bis 79 aus.

Den Beweis bringen folgende Programmzeilen:

```
10 PRINT CHR$(147)
20 PRINT TAB(238) "AAAAAA" POS(0)
30 PRINT TAB(238) "B" POS(0)
40 PRINT SPC(40) POS(0)
```

Nach den beiden TAB-Befehlen erscheinten die Positionsnummern 44 und 39. Und nach dem SPC-Befehl für 40 Leerstellen erscheint die Positionsnummer 0, denn diese logische Zeile ist gerade 1 Bildschirmzeile lang, so daß die nächste Positionsnummer 0, das heißt, der Anfang der nächsten logischen Zeile ist.

Der POS-Befehl kann natürlich gut für eine Abfrage verwendet werden nach dem Muster:

```
240 IF POS(0) > 25 THEN ......
```

Basic-Befehl Nr. 34 POS (Argument)
- er liefert die aktuelle Position des Cursors auf dem Bildschirm innerhalb einer »logischen« Zeile, das heißt Werte von 0 bis 79
- wird kein Cursor angezeigt, dann bezeichnet er die Position hinter dem zuletzt berabeiteten Zeichen
- das Argument in der Klammer hinter dem POS-Befehl ist ein »Dummy«-Argument; es muß vorhanden sein, kann jedoch irgendeinen beliebigen Wert haben

Ich habe bei der Aufzählung der Vorteile, welche die Unterprogramm-Technik attraktiv machen, erwähnt, daß es sinnvoll ist, sich eine Bibliothek von nützlichen Unterprogrammen anzulegen, die dann immer wieder in andere Programme eingebaut werden können. Nun, das klingt gut und einleuchtend, aber wie hängt man ein Unterprogramm, das in der »Bibliothek« – sprich auf Band oder Diskette – steht, an ein Hauptprogramm im Computer?

Gute Frage, kann ich nur sagen, denn das Basic der Commodore-Computer kennt leider keinen Befehl dafür. Es gibt ihn, bei anderen Computern oder bei Befehlserweiterungsprogrammen. Er heißt MERGE oder manchmal auch APPEND, was mit »Verschweißen« oder »Anhängen« übersetzt werden kann. Wir müssen ihn in Basic nachvollziehen, was aber nicht schwer ist.

LEKTION 26: Programme zusammenbinden

Wie gesagt, MERGE ist kein Basic-Befehl, sondern ein Kochrezept.

Ich schreibe Ihnen zuerst das Kochrezept auf und erkläre anschließend, was dabei vorgeht. Es werden folgende Schritte gemacht:

1. Ein Programm steht im Speicher (RAM) des Computers. Ein zweites Programm, das auf Band oder Diskette steht, soll dazugeladen werden.

2. Da das Programm auf Band oder Diskette (2. Pro-

gramm) hinter das Programm im RAM (1. Programm) gehängt wird, ist es empfehlenswert, daß das 2. Programm höhere Zeilennummern als das 1. Programm hat.

3. Die folgende Zeile direkt eingeben und mit <RETURN> abschließen:

```
→ POKE 43,PEEK(45)-2:POKE 44,PEEK(46)
```

4. Jetzt wird das 2. Programm ganz normal mit LOAD in den Computer geladen

5. die folgende Zeile direkt eingeben und mit <RETURN> abschließen:

```
→ POKE 43,1:POKE 44,8
```

Das ist alles; jetzt steht das 1. Programm und das 2. Programm aneinandergehängt im Speicher des C64.

Zur Erklärung dieses Kochrezeptes muß ich Sie an das Speicherbild in Lektion 21 erinnern, in dem ich die Speicheraufteilung dargestellt habe. Wie dort gezeigt, beginnt der für Programme verfügbare RAM-Speicher ab Adresse 2048 und geht bis 40959.

In diesem Bereich stehen nun die Zeilen eines Programms, alle Namen und augenblicklichen Werte von Variablen, Felder und schließlich auch alle Texte von String-Variablen. Um ein Chaos zu vermeiden und um die Suchzeiten nach Variablenwerten möglichst kurz zu halten, ist dieser Speicherteil in Bereiche unterteilt.

Der Arbeitsspeicher ist unabhängig von seiner durch Speichererweiterung veränderbaren Größe in vier Blöcke und einen Freiraum eingeteilt.

- Block 1 beginnt immer an 2048 und enthält die Zeilen (Text) eines eingegebenen Programms. Sein oberes Ende ist variabel, es hängt von der Länge des Programms ab. Deshalb habe ich diese Grenze als Strich mit einem Pfeil dargestellt.
- Block 2 schließt sich direkt an Block 1 an und enthält die numerischen Variablen und die Namen der String-Variablen. Sein Anfang ist identisch mit dem Ende des 1. Blocks. Deshalb ist dieser Anfang ebenfalls variabel. Dadurch ist aber auch das Ende von Block 2 variabel. Wenn zum Beispiel in ein Programm weitere Zeilen eingefügt werden, rutschen alle mit Pfeil markierten Grenzen der Speicherbereiche nach oben.
- Block 3 enthält alle Felder.
- Block 4 beginnt am obersten Ende des Arbeitsspeichers und hängt von der Ausbaustufe des Speichers ab.

  Er enthält alle Texte der String-Variablen, auch die der Felder. Je nach Länge beziehungsweise Menge der Texte bewegt sich die Grenze des 4. Blocks nach unten, solange bis sie die Grenze des 3. Blocks (die ja nach oben wandert) trifft. Dann ist der Speicher voll, und der Computer meldet OUT OF MEMORY.

Jetzt kommt der Aspekt der Speicheraufteilung, der für unser MERGE-Kochrezept wesentlich ist. Es ist sicher verständich, daß der Computer, oder besser gesagt seine »Betriebsleitung«, jederzeit wissen muß, wo gerade die verschiedenen Grenzen der Blöcke liegen. Das Betriebssystem besitzt dazu im Systemspeicher (von Adresse 0 bis 1023) einige Speicherzellen, in denen der jeweilige Adressenstand der Blockgrenzen gespeichert ist.

Diese Adressen treten immer paarweise auf. Das liegt daran, daß in einer Speicherzelle allein stets nur Zahlen von 0 bis 255 gespeichert werden können. Das ergäbe 256 Adressen – wir brauchen aber mehr. Wenn man zwei Zellen sozusagen aneinanderhängt, erreicht man dadurch 256*256=65536 Adressen. Diese doppelte Wortlänge – auch Low-/High-Byte-Darstellung genannt – erfordert natürlich einen Rechenvorgang, um aus dem Inhalt von zwei Speicherzellen die dadurch repräsentierte Adresse zu erhalten.

Wie und warum das so ist, erkläre ich ein paar Absätze weiter unten im Abschnitt »Die Low-/High-Byte-Darstellung«. Hier ist nur die Umrechnungsformel interessant. Steht in den beiden Speicherzellen X und Y eine Adresse, wird diese ausgerechnet mit:

Adresse = X + 256*Y

Das MERGE-Konzept wollen wir dadurch ausprobieren, daß wir vor das Listing 8 (das Spiel mit Zeitbegrenzung) das Listing 6, nämlich die farbige Umrandung setzten und so ein Programm mit Vorspann daraus machen.

In dem Speicherzellenpaar 43/44 steht also die Anfangsadresse des Arbeitsspeichers. Wir prüfen das nach mit:

```
→ X=PEEK(43):Y=PEEK(44)
  PRINT X:PRINT Y
  PRINT X+256*Y
```

Das ganze geht natürlich viel kürzer in einer einzigen Zeile:

```
→ PRINT PEEK(43) + 256 * PEEK(44)
```

Wir erhalten beim C64 die Zahl 2049 als Speicheranfang. Entsprechend steht im Speicherzellenpaar 45/46 die Adresse des Endes von Block 1.

Laden Sie bitte Listing 6 in den Speicher und überprüfen das Blockende mit:

```
→ PRINT PEEK(45) + 256 * PEEK(46)
```

Bei mir erscheint da die Zahl 2252, ich hoffe, bei Ihnen auch.

Beim MERGE-Kochrezept führen wir das Betriebssystem des C64 hinters Licht. Mit den Befehlen:

```
→ POKE 43,PEEK(45)-2:POKE 44,PEEK(46)
```

schreiben wir in die Speicherzelle 43 diejenige Zahl, um 2 vermindert, die in Speicherzelle 45 steht. Das gleiche machen wir mit der Speicherzelle 44.

Dadurch verschieben wir mutwillig die Anfangsadresse des Arbeitsspeichers an das Ende des Blocks 1. Da die letzten 2 Speicherzellen hinter einem Programm zur Kennzeichnung des Endes immer nur Nullen enthalten, ziehen wir die 2 ab.

Der Computer glaubt nun, der Arbeitsspeicher beginne bei Adresse 2550 – und ab da lädt er jetzt das Listing 8, ohne Listing 6 zu berühren. Danach allerdings müssen wir die Anfangsadresse wieder an ihren ursprünglichen Platz schieben mit:

```
→ POKE 43,1:POKE 44,8
```

denn 1+256*8 ergibt 2049, was vorher ja dort stand.

Wenn Sie jetzt LISTen, dann stehen beide Programme als ein neues größeres Programm im Speicher und es kann – unter neuem Namen – gespeichert werden.

Speicherzellenpaare wie 43/44, 45/46 nennen wir Zeiger, weil sie auf eine andere Adresse zeigen. Das Hantieren mit Zeigern ist eine sehr reizvolle Sache. Ich bin sicher, Sie werden noch oft damit konfrontiert. Es gibt viele Artikel darüber, ich will mich in diesem Anfängerkurs auf das bisher Gesagte beschränken – es soll ja auch nur eine Einführung sein.

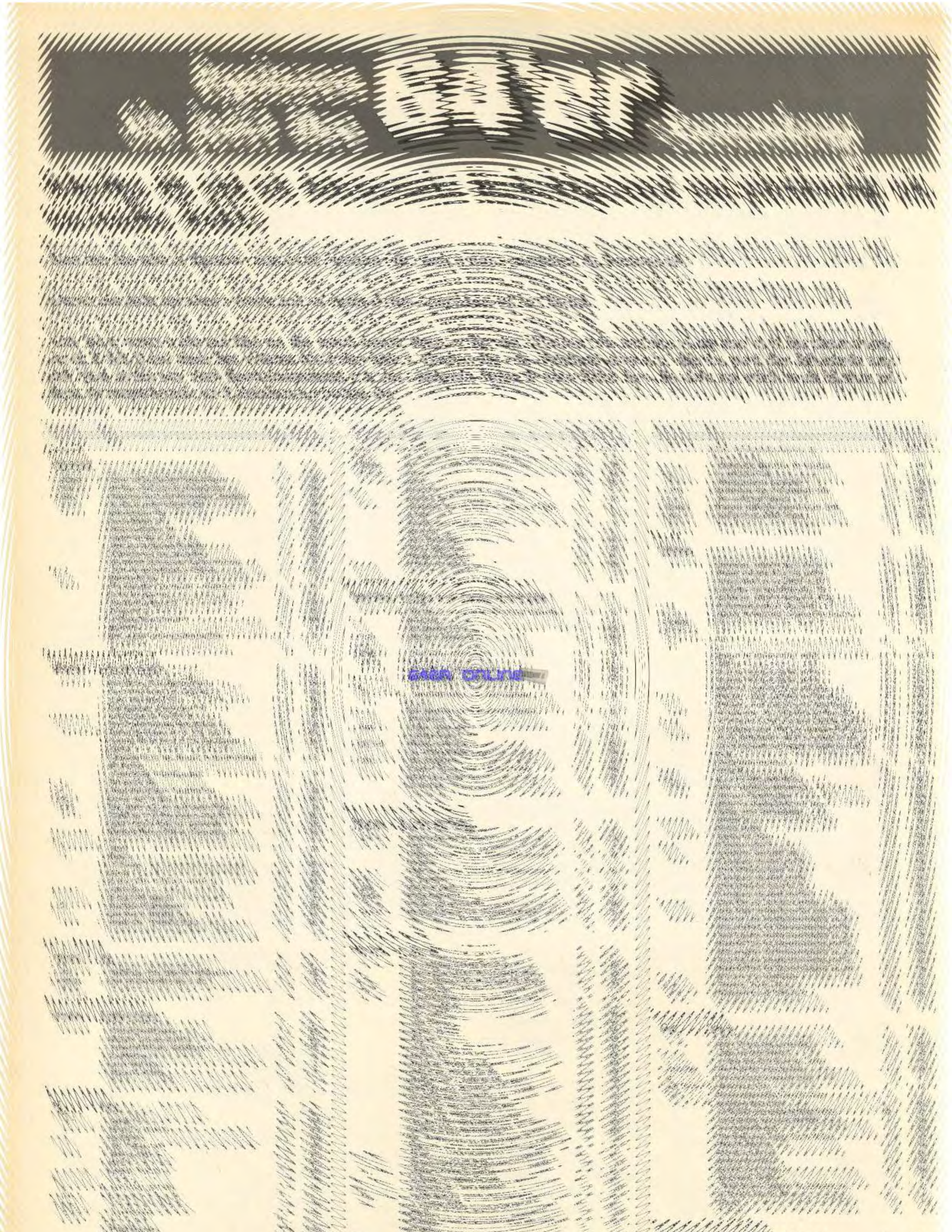
Doch halt, die doppelte Wortlänge muß ich noch erklären.

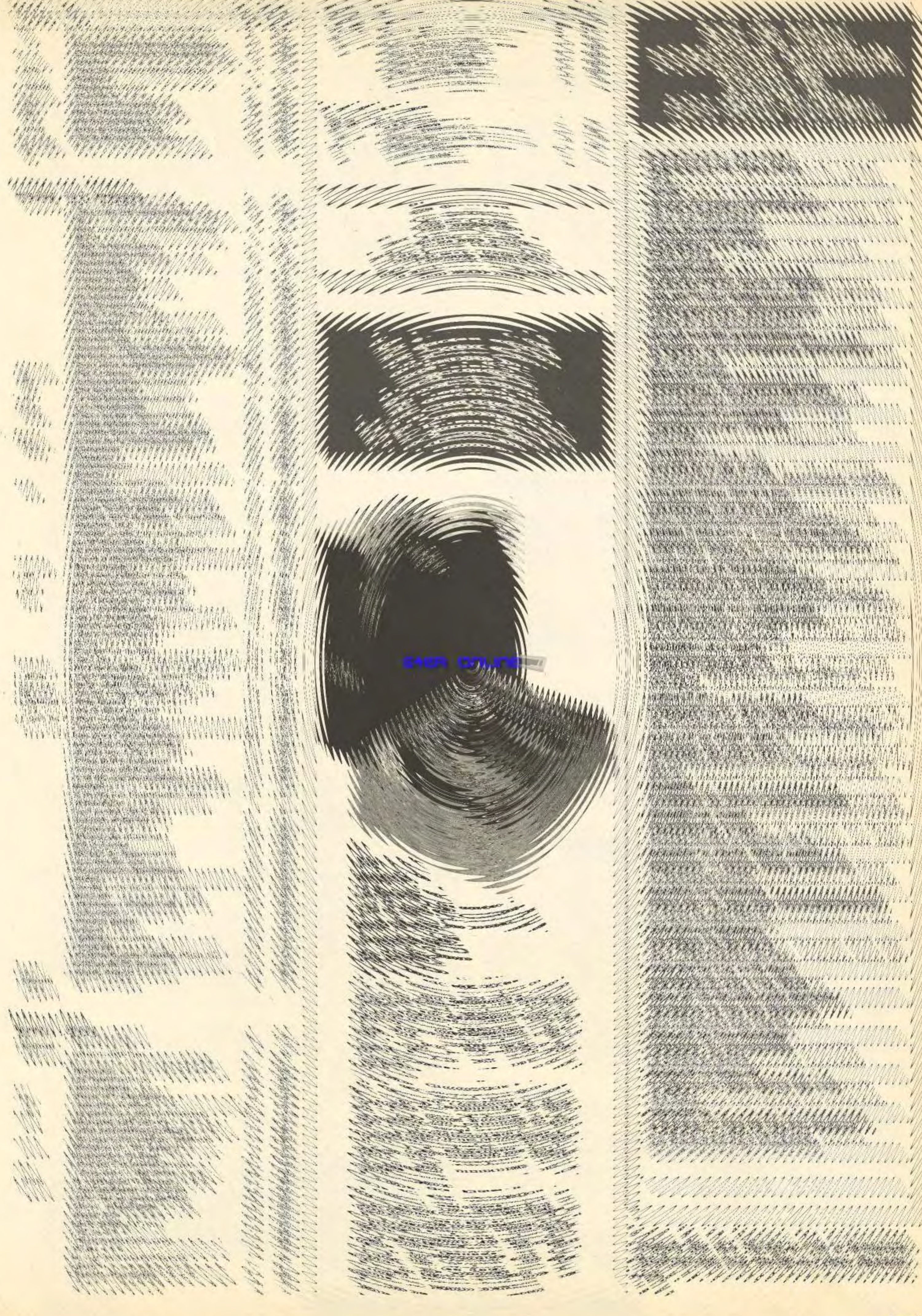
LEKTION 27: Die Low-/High-Byte-Darstellung

Ich setze voraus, daß Sie das duale Zahlensystem mit seinen Nullen und Einsen kennen, und daß Sie dadurch wissen, was ein Bit ist, außerdem, wie man Dualzahlen in Dezimalzahlen und umgekehrt umwandelt.

Sollte dies jedoch nicht der Fall sein, dann lesen Sie bitte im 64'er Sonderheft 5/1986 auf Seite 90 den Teil Bits und Bytes – er gibt eine kleine Einführung.

Eine Speicherzelle des C64 hat eine Länge von 8 Bit = 1 Byte.





Mit diesen 8 Bit können Zahlen von 0 bis 255 dargestellt werden. Zur Darstellung von Zahlen über 255 verwenden wir die Low-/High-Byte-Methode.

Wir hängen einfach 2 Speicherzellen zusammen, mit deren 16 Bit wir Zahlen bis maximal 65535 darstellen können. Die maximale Zahl 65535 ist übrigens auch die höchste Adresse des gesamten Speichers - was natürlich kein Zufall ist.

Ich will Ihnen jetzt zeigen, wie eine Dezimalzahl auf zwei 8-Bit-Speicherzellen verteilt wird und umgekehrt: Wie aus zwei Bytes eine Dezimalzahl gebildet wird.

Schauen Sie sich das folgende Beispiel an:

DEZIMAL	47491	
DUALZAHL	1011 1001	1000 0011
HIGH BYTE	185	
LOW BYTE		131

Wir gehen von der Dezimalzahl 47491 aus. Ihre duale Darstellung mit 16 Bit - 1011100110000011 - teilen wir einfach in der Mitte und erhalten damit zwei neue Dual-Zahlen mit je 8 Bit = 1 Byte. Das linke Byte nennen wir High-Byte, da es den höheren Teil der Gesamtzahl darstellt. Das rechte Byte heißt entsprechend Low-Byte. Jedes der beiden Bytes kann für sich allein in einer Speicherzelle untergebracht werden, in der natürlich dann der dezimale Wert des Bytes steht. Zur Umrechnung der Low-/High-Bytes empfehle ich folgende Kochrezepte:

Dezimal in Low-/High-Byte

47491:256=185 (High-Byte), Rest 131 (Low-Byte)

Der Rest fällt bei der Division per Hand automatisch an. Mit dem (Taschen)Rechner erhält man den Rest durch:

185*256-47491=-131

Low-/High-Byte in Dezimal

High-Byte * 256 + Low-Byte = Dezimal

185 * 256 + 131 = 47491

Eine wichtige Regel gilt es noch zu beachten: Der Mikroprozessor des C64 verlangt, daß immer das Low-Byte vor dem High-Byte kommen muß. Die Zahl wird sozusagen von rechts nach links gelesen, in unserem Beispiel 131, 185.

LEKTION 28: Dateien (Files)

In Lektion 20 haben wir ein Programm gebaut, in dem Variablenwerte in ein Feld gespeichert wurden. Wir waren damals noch nicht in der Lage, diese Feldvariablen auf Band oder Diskette abzuspeichern. Ich habe Sie damals gebeten, das Programm-Fragment abzuspeichern, bis wir soweit wären, mehr vom Speicher und vom Speichern zu wissen.

Jetzt sind wir soweit. Ein Blick auf Bild 7 macht es uns deutlich. Mit den Befehlen LOAD speichern wir nur den Inhalt vom 1. Block ab. Die Feldvariablen stecken aber im 3. Block!!

28.1. Der Unterschied zwischen Programm und Datei (File)

Der Computer unterscheidet glücklicherweise beim Speichern auf externe Speichergeräte (Datasette, Diskette) zwischen Daten aus dem 1. Block und den Blöcken 2, 3 und 4. Daten aus den letzteren nennen wir eine *Datei* oder auf englisch *File*.

Ein Programm besteht aus einer Ansammlung von numerierten Zeilen. Jede Zeile enthält Anweisungen an den Computer. Diese Anweisungen werden nach RUN zeilenweise ausgeführt, in Reihenfolge der aufsteigenden Zeilennummern.

Eine Datei (File) besteht aus rohen Daten, wie Angaben in einem Adreßbuch, ohne jede Angabe, was damit gemacht werden soll. Der bekannte Computerjournalist Richard Mansfield hat zur Erklärung dieses Unterschieds einmal das einleuchtende Beispiel einer Dose mit Tomatensuppe gebracht. Der Aufkleber der Dose enthält sowohl eine Datei als auch ein Programm.

Die Datei lautet:
Wasser, Tomaten, Salz,
Monosodiumglutamat, Farbstoff,
Oleoresin

Das Programm lautet:
1. Vorsichtig öffnen
2. Inhalt in einen Topf leeren
3. eine Tasse Wasser hinzufügen
4. auf kleiner Flamme erhitzen
5. ungewürzt servieren

Ich bin sicher, Sie sehen jetzt den Unterschied.

Programme werden mit SAVE auf Band und auf Diskette gespeichert.

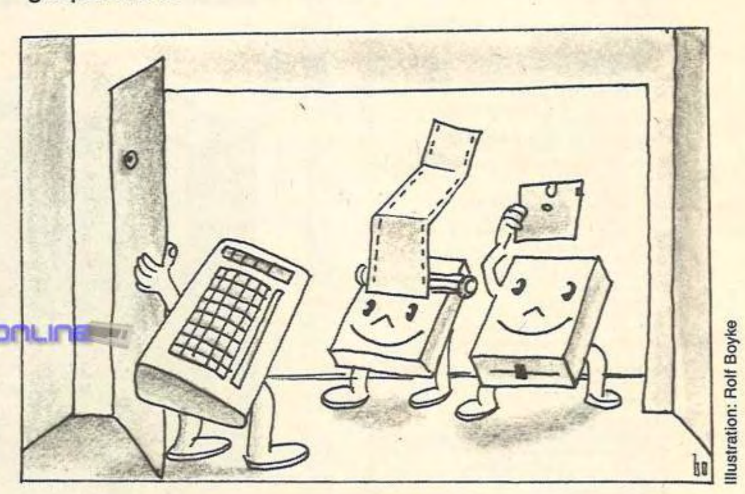


Illustration: Rolf Boyke

Dateien werden mit einer eigenen Befehlskombination OPEN, PRINT# und CLOSE sowohl auf Band, als auch auf Disketten gespeichert.

Genau das wollen wir jetzt lernen.

28.2. Dateien (Files) speichern

Wenn man eine Datei laden oder speichern will, muß zu allererst eine Verbindung zwischen Computer und dem externen Speichergerät hergestellt werden. Ich meine damit nicht das Verbindungskabel, das brauchen wir bei Speichern und Laden von Programmen ja auch. Ich meine hier eine Anweisung an den Computer, sich für den Transfer von Daten über das externe Gerät bereitzuhalten.

Man nennt das auch:

»eine Datei eröffnen«.

Es ist vergleichbar mit dem Herausziehen und Öffnen eines Karteikastens, um Zugriff zu den Daten auf den Karteikarten zu haben.

Der Befehl dazu heißt

OPEN

was natürlich »Öffnen« heißt.

Basic-Befehl Nr. 35 OPEN

- der OPEN-Befehl öffnet eine Datei zum Schreiben oder Lesen. Mit diesem Befehl wird automatisch ein Pufferspeicher angelegt, mit dem die verschiedenen Arbeitsgeschwindigkeiten der externen Speichergeräte ausgeglichen werden können.

- hinter dem OPEN-Befehl stehen bis zu 6 weitere Angaben, die durch Kommata voneinander getrennt sein müssen. Diese Angaben sind:
  OPEN Dateinummer,Gerätenummer, Sekundäradresse, "Dateiname,Typ,Modus"

Die zusätzlichen Angaben hinter OPEN bedürfen einer Erklärung. Sie sind sicher für den Anfänger verwirrend, und wir brauchen sie jetzt auch gar nicht alle. Man findet jedoch selten eine komplette Übersicht, daher mache ich die Aufstellung vollzählig – für später.

Datei-Nummer
ist die Kennzeichnung der eröffneten Datei. Sie kann eine Zahl zwischen 1 und 255 sein. Sie ist wichtig und dient als Referenz beim Umgang mit dieser bestimmtem Datei; dies um so mehr, als gleichzeitig bis zu 10 verschiedene Dateien geöffnet sein können.

Gerätenummer
ist eine Kennzahl dafür, mit welchem externen Gerät die Verbindung hergestellt wird. Dabei gilt:
0 Tastatur
1 Kassettenrecorder (Datasette)
3 Bildschirm
4 Drucker 1
5 Drucker 2, meistens Plotter
8 Diskettenlaufwerk 1
9 Diskettenlaufwerk 2
4 bis 127 Geräte, die am seriellen Bus angeschlossen sind
128 bis 255 Geräte über seriellen Bus, mit einem Zeilenvorschub nach dem Zeilenende.

Diese Angaben sagen Ihnen jetzt noch nichts. Übrigens: Wenn die Gerätenummer weggelassen wird, stellt der Computer automatisch die 1 für Datasette ein.

Sekundär-Adresse
kann eine Zahl zwischen 0 und 255 sein. Diese Zahl hat bei den verschiedenen externen Geräten unterschiedliche Bedeutung.

Tastatur		keine Wirkung
Kassettenrecorder	0	vom Band lesen
	1	auf Band schreiben
	2	auf Band mit »Bandende«-Markierung schreiben
Bildschirm		keine Wirkung
Drucker	0 bis 10	die Funktionen sind bei den Druckerfabrikaten zum Teil unterschiedlich, bitte im Drucker-Handbuch nachsehen
Diskettenlaufwerk		
	0 und 1	vom Betriebssystem reserviert
	2 bis 14	reserviert einen numerierten Datenkanal, bis zu drei gleichzeitig
	15	reserviert einen Befehls-Kanal

Datei-Name
steht immer zwischen Anführungszeichen. Er kann bei der Datasette auch weggelassen werden.

Datei-Typ
wird nur bei Diskettenoperationen verwendet und steht innerhalb der Anführungszeichen hinter dem Datei-Namen; er bezeichnet die folgenden Typen:
S sequentielle Datei
U User-Datei
P Programm-Datei
R relative Datei

Wir werden uns im Rahmen dieses Kurses nur mit sequentiellen Dateien befassen, erstens weil mit der Datasette nur dieser Typ möglich ist und zweitens, weil er leichter zu verstehen ist.

Modus
wird nur bei Diskettenoperationen verwendet und legt fest, wie der Datenkanal genutzt werden soll:
W Schreiben einer Datei (Write)
R Lesen einer Datei (Read)
A Verlängern einer sequentiellen Datei (Append)
M Lesen einer nicht geschlossenen Datei
Nun bitte keine Panik!

In diesem Kurs brauchen wir nur Bruchteile daraus. Aber immerhin ist das alles ein guter Fingerzeig, daß speziell Diskettenoperationen sehr raffiniert sein können.

Zurück zur Anwendung des OPEN-Befehls. Mit:

```
→ 10 OPEN 1,1,1,"TEXT"
```

eröffnen wir also die Datei "TEXT" auf der Datasette.

Für die Diskette lautet derselbe Befehl:

```
→ 10 OPEN 1,8,3,"TEXT,S,W"
```

S steht für sequentielle Datei, W für Schreiben.

Um jetzt Daten auf das externe Gerät schreiben zu können, verwenden wir eine Variante des PRINT-Befehls, er heißt

PRINT#

Hinter PRINT# muß immer dieselbe Datei-Nummer stehen, die beim OPEN-Befehl verwendet worden ist – sie ist die Referenz. Danach folgt die zu schreibende Variable.

```
→ 20 PRINT#1,"FUSSBALL"
  30 PRINT#1,"TENNIS"
```

Damit haben wir die zwei Wörter abgesendet. Hinter einem PRINT#-Befehl darf immer nur eine Variable oder ein String stehen, damit die die Daten in der Datei einen Abstand haben. Als nächstes muß die eröffnete Datei wieder geschlossen werden. Entsprechend zum Eröffnungsbefehl lautet der Schließbefehl CLOSE. Er wird mit der Nummer der Datei versehen.


```
→ 40 CLOSE 1
  50 END
```

Wenn Sie diese vier Zeilen laufen lassen, fordert die Datasette zum Drücken der RECORD- und PLAY-Tasten auf, das Diskettenlaufwerk läuft sofort los und die Datei ist gespeichert.

Dasselbe machen wir jetzt für ein anderes Ausgabegerät, nämlich für den Drucker – mit der Gerätenummer 4.

```
→ 10 OPEN 1,4
  20 PRINT#1,"FUSSBALL"
  30 PRINT#1,"TENNIS"
  40 CLOSE 1
```

Zeile 10 bewirkt, daß die beiden Wörter jetzt vom Drucker ausgedruckt werden.

Basic-Befehl Nr. 36 PRINT#
- der PRINT#-Befehl sendet Daten einer Datei auf Band, Diskette oder Drucker. Maximal können mit einem PRINT#-Befehl 255 Zeichen gesendet werden
- dem PRINT#-Befehl muß dieselbe Dateinummer folgen wie dem OPEN-Befehl, der die Datei eröffnet hat
- es können numerische und String-Variable sowie Feld-Variable gesendet werden. Wichtig ist, daß die einzelnen Variablen durch das RETURN-Zeichen voneinander getrennt sind. RETURN kann entweder durch separate PRINT#-Befehle oder durch CHR$(13) erzeugt werden

Basic-Befehl Nr. 37 CLOSE
- dieser Befehl schließt eine eröffnete Datei
- der CLOSE-Befehl wird lediglich mit der Nummer der Datei versehen, die er schließen soll
- jede Datei muß unbedingt geschlossen werden, da andernfalls keine Endmarkierung hinter die letzte Dateneintragung geschrieben wird. Dadurch können die letzten Daten verlorengehen. In dem Inhaltsverzeichnis (Directory) einer Diskette ist eine nicht geschlossene Datei mit dem Stern »*« gekennzeichnet.

28.3 Dateien (Files) laden

Jetzt kommt der andere Teil – eine auf Band oder Diskette gespeicherte Datei in den Computer laden oder lesen. Die Datei muß wieder eröffnet werden, diesmal zum Lesen.

Für die Datasette lautet der Befehl:

```
100 OPEN 1,1,0,"DATEI"
```

Für die Diskette dagegen muß es heißen:

```
100 OPEN 1,8,3,"DATEI,S,R"
```

Um Daten einer Datei zu lesen, gibt es gleich zwei Befehle, die Sie in ähnlicher Form schon kennen, nämlich INPUT# und GET#.

Zuerst wenden wir den INPUT#-Befehl an.

In unserer Datei haben wir zwei Strings gespeichert. Mit folgender Zeile lesen wir sie:

```
110 INPUT#1,A$,B$
120 PRINT A$,B$
130 CLOSE 1
```

In Zeile 120 trenne ich das Ausdrucken von A$ und B$ mit einem Komma. Mit dem Klebeeffekt des Semikolon würde nämlich das Wort FUSSBALLTENNIS ausgedruckt werden. Probieren Sie das kleine Leseprogramm mit RUN 100 aus.

Das ganze Listing für Schreiben und Lesen einer String-Datei sieht so aus:

```
10 OPEN 1,1,1,"TEXT"          ...für Band
10 OPEN 1,8,3,"TEXT,S,W"      ...für Diskette
20 PRINT#1,"FUSSBALL"
30 PRINT#1,"TENNIS"
40 CLOSE 1
50 END
60 :
100 OPEN 1,1,0,"DATEI"        ...für Band
100 OPEN 1,8,3,"DATEI,S,R"    ...für Diskette
110 INPUT#1,A$,B$
120 PRINT A$,B$
130 CLOSE 1
```

Für eine Datei mit numerischen Variablen sieht das Programm fast gleich aus:

```
200 OPEN 1,1,1,"ZAHLEN"       ...für Band
200 OPEN 1,8,3"ZAHLEN,S,W"    ...für Diskette
210 PRINT#1,35
220 PRINT#1,14
230 PRINT#1,753
240 PRINT#1,2
250 CLOSE 1
260 END
270 :
300 OPEN 1,1,0,"ZAHLEN"       ...für Band
300 OPEN 1,8,3,"ZAHLEN,S,R"   ...für Diskette
310 INPUT#1,A,B,C,D
320 PRINT A;B;C;D
330 CLOSE 1
```

Bei Zahlen dürfen wir in Zeile 320 ruhig Semikolons verwenden, da Zahlen von selbst Trennungsabstände haben.

Basic-Befehl Nr. 38 INPUT#
- der INPUT#-Befehl liest, wie GET#, Daten einer mit OPEN eröffneten Datei externer Eingabegeräte
- hinter INPUT# muß die Datei-Nummer des OPEN-Befehls stehen
- während GET# Zeichen für Zeichen liest, holt INPUT# ganze Datengruppen
- hinter einem INPUT#-Befehl können mehrere, durch Kommata getrennte Variable stehen
- ein String darf nicht länger als 80 Zeichen sein
- INPUT# kann nicht im Direkt-Modus verwendet werden

Wir wollen noch die Wirkungsweise des GET#-Befehls ausprobieren. Ersetzen Sie im ersten Programm in Zeile 110 das INPUT# durch GET#:

```
110 GET#1,A$,B$
```

Nach RUN 100 erhalten wir den weit auseinandergezogenen Ausdruck der Buchstaben F und U. So geht es also nicht – GET# nimmt immer nur ein Zeichen.

Wir müssen GET# wiederholt einsetzten, am besten mit einer Schleife.

```
110 GET#1,A$
120 PRINT A$;
125 GOTO 110
```

Jetzt erhalten wir die beiden Wörter FUSSBALL und TENNIS, aber die Schleife stoppt nicht. Wir prüfen daher, wann das letzte Zeichen erscheint und CLOSEn dann die Datei. Dazu müssen wir im Eingabeteil (Zeile 30) hinter dem letzten String ein einzigartiges, erkennbares Zeichen einsetzen. Ich habe einfach hinter Tennis ein Leerzeichen gesetzt.

Leider müssen wir dadurch die Datei neu eingeben, am besten mit einem anderen Namen in Zeile 10 und 100. Oder aber Sie löschen die alte Datei »TEXT«.

```
→ 30 PRINT#1,"TENNIS "
  123 IF A$=" " THEN 130
```

Das sind die geänderten Zeilen.

Basic-Befehl Nr. 39 GET#
- der GET#-Befehl funktioniert genauso wie der normale GET-Befehl: Er liest einzelne Zeichen einer geöffneten Datei von einem Eingabegerät. Die Anzahl der Zeichen (Länge der Strings) ist beliebig
- hinter GET# muß die Datei-Nummer des OPEN-Befehls stehen
- GET# kann nicht im Direkt-Modus verwendet werden

Das wiederholte Schreiben der einzelnen PRINT#-Befehle ist natürlich sehr lästig. Besser geht das mit einer FOR-NEXT-Schleife, wobei die zu speichernden Daten entweder direkt per normalen INPUT-Befehl eingegeben oder mit READ aus gespeicherten DATA-Zeilen geholt werden. Ich zeige das an unserer TEXT-Datei in der GET-Version:

```
→ 10 OPEN 1,8,3,"TEXT,S,W"      ...für Diskette
  10 OPEN 1,1,1,"TEXT"          ...für Band
  20 FOR I=1 TO 3
  30 READ A$
  40 PRINT#1,A$
  50 NEXT I
  60 CLOSE 1
  70 DATA FUSSBALL,TENNIS,*
  80 END
  90 :
  100 OPEN 1,8,3,"TEXT,S,R"     ...für Diskette
  100 OPEN 1,1,0,"TEXT"         ...für Band
  110 GET#1,A$
  120 IF A$="*" THEN 130
  123 PRINT A$;
  125 GOTO 110
  130 CLOSE 1
```

Ich habe die Zeile 120, die auf Datei-Ende prüft, leicht geändert. Sie prüft jetzt auf ein abschließendes Zeichen »*«, das ich als letzte Eintragung in die neue DATA-Zeile 70 geschrieben habe. READ und PRINT#1 erfolgen nun innerhalb einer Schleife zwischen Zeile 20 und 50.

Sie müssen leider wieder die Datei TEXT entweder löschen (Diskette) oder überschreiben (Band). Mit RUN läuft der Schreibteil des Programms bis Zeile 80. Mit RUN 100 starten Sie den Leseteil.

Den abspeichernden Teil (also Zeilen 10 bis 70) können wir natürlich auch für die Druckerausgabe verwenden.

Der OPEN-Befehl lautet dafür:

```
210 OPEN 1,4
```

Ab Zeile 220 können Sie die Zeilen 20 bis 70 vom oberen Programm übernehmen, indem Sie eine 2 vor die »alten« Zeilennummern eingeben.

Das Programm sieht jetzt so aus:

```
210 OPEN 1,4
220 FOR I=1 TO 3
230 READ A$
240 PRINT#1,a$
250 NEXT I
260 CLOSE 1
270 DATA FUSSBALL,TENNIS,*
```

Nach RUN druckt der Drucker die Wörter der DATA-Zeile aus. So einfach ist das.

28.4. Kommandogewalt an das externe Gerät

Basic bietet die Möglichhkeit an, mit »normalen« Befehlen die angeschlossenen Geräte anzusteuern. Das beste Beispiel dafür ist der Befehl LIST. Wenn er normalerweise das im Programmspeicher des C64 stehende Programm auf dem Bildschirm auslistet, dann soll es möglich sein, mit LIST das Programm auch auf dem Drucker auszudrucken.

Der Befehl dazu ist vom englischen Wort »Command« abgeleitet und heißt

CMD Dateinummer

In Anwendung sieht das, als Direktbefehl eingegeben, so aus:

→ `OPEN 1,4:CMD 1:LIST`

Jetzt gilt der LIST-Befehl nicht für den Bildschirm, sondern für den Drucker.

Um die Kommandogewalt wieder dem Bildschirm zu übertragen, muß die Verbindung zum Drucker gelöst werden mit:

→ `PRINT#1:CLOSE 1`

Der CMD-Befehl kann auch mit einem Kommentar (wie beim INPUT-Befehl) versehen werden, den er im angesprochenen Gerät »ausdruckt«. Im obigen Beispiel angewendet:

→ `OPEN 1,4:CMD 1,"LISTING-NAME":LIST`

Dadurch wird zu Beginn des Listings die Überschrift LISTING-NAME ausgedruckt.

Ein Programm im Speicher kann auch als sequentielle Datei auf die Diskette abgespeichert werden:

→ `OPEN 1,8,2,"PROGRAMM-NAME,S,W"`

→ `CMD 1:LIST`

Jetzt wird das Programm auf der Diskette »ausgeLISTet«, das heißt, es wird sequentiell gespeichert.

→ `PRINT#1:CLOSE 1`

Ausgelesen wird es in der schon beschriebenen Weise.

Nicht nur LIST ist ein sinnvoller Befehl, auch PRINT kann anstelle von PRINT# verwendet werden.

```
10 OPEN 1,4
20 CMD 1
30 PRINT"FUSSBALL"
40 PRINT"TENNIS"
50 CLOSE 1
```

Damit werden die beiden Wörter FUSSBALL und TENNIS direkt auf dem Drucker ausgedruckt – mit PRINT ohne # !

Basic-Befehl Nr. 40 CMD

- er muß immer mit derselben Dateinummer versehen sein, mit der eine Datei durch OPEN eröffnet worden ist.
- die dem CMD-Befehl folgenden Befehle (z.B. LIST oder PRINT) wirken dann auf das angewählte Gerät.
- CMD kann, durch ein Komma getrennt und mit Anführungszeichen versehen, mit einem Kommentar versehen werden, der dann ausgedruckt oder gespeichert wird:

CMD 1,"KOMMENTAR"

- die Wirkung von CMD kann auf 3 Arten aufgehoben werden:
  - CMD 3 (Gerätenummer des Bildschhirms)
  - PRINT#
  - STOP+RESTORE-Taste

LEKTION 29: Ein Lied als Datei speichern

Zum Schluß der Datei-Betrachtungen will ich mein Versprechen einlösen, das ich in Lektion 20 bei den zweidimensionalen Feldern gemacht habe. Ich will Ihnen nämlich zeigen, wie man Werte, die wir in Felder gelegt haben, abspeichern, wieder laden und variieren kann.

Dazu verwenden wir Listing 5, unser Programmbeispiel für zweidimensionale Felder.

Wir wollen schrittweise vorgehen.

→ Laden Sie Listing 5 ein

→ Fügen Sie folgende Zeilen dazu:

```
500 REM---- DATEI ABSPEICHERN -----
505 :
510 OPEN 1,8,4,"LITERATUR,S,W"
520 FOR S=1 TO D
530 FOR K=1 TO 6
540 PRINT#1,A$(S,K)
550 NEXT K:NEXT S
560
670 CLOSE 1
```

Dieser Programmteil speichert, wie Sie nicht nur aus dem Titel, sondern auch aus dem W (write) nach dem OPEN-Befehl sehen können, das gesamte Feld A$ als Datei auf Diskette ab. Für die Datasette muß lediglich Zeile 510 geändert werden:

```
510 OPEN 1,1,1,"LITERATUR"
```

→ `RUN`

Damit lassen wir das Programm laufen, wodurch die Datensätze der DATA-Zeilen in das Feld eingelesen werden. Der nachfolgende Suchteil des Programms wird jetzt nicht weiter gebraucht und kann durch ein falsches Suchwort und durch »N« bei der Wiederholungsfrage abgebrochen werden.

→ `GOTO 500     (nicht RUN 500!)`

Genau wie RUN 500 setzt der Direktbefehl GOTO 500 das Programm ab Zeile 500 fort, allerdings ohne die Variablen auf Null zurückzusetzen. Das ist wichtig, damit D seinen letzten, dem aktuellen Stand der Anzahl von Datensätzen entprechenden Wert behält. Mit den Zeilen 500 bis 570 wird das Feld als Datei abgespeichert.

→ Löschen Sie alle DATA-Zeilen

Wir brauchen sie nicht mehr, denn die Datensätze sind ja als Datei gespeichert.

Um die Datei wieder in den Computer laden zu können, müssen wir den READ-Teil des Programm (Zeilen 110 bis 170) umschreiben. Ich schreibe nur die Zeilen, die sich ändern:

```
→ 115 OPEN 1,8,3,"LITERATUR,S,R"
  (115 OPEN 1,1,0"LITERATUR" für Datasette)
  140 INPUT#1,A$(D,K)
  150 IF A$(D,K)= .....usw..:GOTO 180
  180 CLOSE 1
```

Um das Einlesen der Datei »LITERATUR« ausprobieren zu können, müssen wir das noch im Speicher stehende Feld mit den Datensätzen löschen, am besten mit Aus- und Einschalten des Computers. Vorher aber speichern Sie auf alle Fälle das unfertige Programm, am besten als Listing 10 A.

→ Computer aus- und wieder einschalten

→ Listing 10 A wieder LOADen und mit RUN laufen lassen

Der Suchvorgang wird Ihnen zeigen und beweisen, daß das Feld tatsächlich als Datei mit dem Namen »LITERATUR« auf der Diskette (Kassette) gespeichert ist.

FAZIT
1. der Direktbefehl RUN 500 setzt alle Variablen auf den Wert 0 zurück und beginnt das Programm ab Zeile 500
2. der Direktbefehl GOTO 500 setzt das Programm ebenfalls ab Zeile 500 fort, ohne aber die Variablen zu verändern

LEKTION 30: Ein vollständiges Datei-Programm

Für ein vollständiges Datei-Programm fehlen dem Listing 10 A noch zwei Dinge:
- ein Menü zur freundlichen Bedienung
- eine Möglichkeit, die Datei zu erweitern

Das Menü soll uns helfen, das ganze Programm, das ja schon fast vollständig ist, zu strukturieren.

Das Menü ist genausoaufgebaut wie das Beispiel in Lektion 24.

```
5 REM---- MENUE -----
7 :
10 PRINT CHR$(147)
15 PRINT SPC(90)"WAEHLEN SIE BITTE AUS"
20 PRINT TAB(202)"(1) DATEN LADEN"
25 PRINT SPC(82)"(2) DATENSAETZE SUCHEN"
30 PRINT SPC(82)"(3) NEUEN TEXT EINGEBEN"
35 PRINT SPC(82)"(4) DATEI ABSPEICHERN"
40 PRINT SPC(82)"(5) ENDE"
45 :
50 PRINT
55 INPUT A
60 ON A GOSUB 100,200,400,500,600
65 GOTO 10
```

Wie gesagt, es ist dasselbe Menü wie vorher, nur der Text ist dem Datei-Programm angepaßt. Auch die Unterprogramm-Zeilennummern in Zeile 60 entsprechen dem Datei-Programm.

Die einzelnen Unterprogramme, soweit sie bereits vorhanden sind, müssen mit RETURN abschließen. Das wollen wir gleich nachholen:

```
→ 190 RETURN
  340 RETURN
  580 RETURN
```

Jetzt fehlen nur noch die Unterprogramme »NEUEN TEXT EINGEBEN« (ab Zeile 400) und »ENDE« (ab Zeile 600)

```
→ 400 REM------ NEUEN TEXT EINGEBEN -------
  410 :
  420 K=1
  430 PRINT K". KATEGORIE":INPUT A$(D,K)
  440 IF K<6 THEN K=K+1:GOTO 430
  450 PRINT:PRINT"NOCH EINE EINGABE (J/N) ?"
  460 GET V$:IF V$="" THEN 460
  470 IF V$="J" THEN D=D+1:GOTO  410
  480 GOTO 500
```

Dieses Unterprogramm ersetzt die Eingabe über DATA-Befehle. Die entscheidende Zeile ist Zeile 430. In ihr wird die Eingabe eines kompletten Datensatzes verlangt, wobei die 6 Kategorien numeriert einzeln aufgerufen werden. Diese Eingabe wird sofort einer Feldvariablen A$(D,K) zugewiesen.

Das alles ist nur möglich, weil die Variable D, welche die Anzahl der Datensätze angibt, auch nach dem Suchprogramm wieder auf dem Endwert steht. Sie braucht bei der ersten Eingabe eines neuen Datensatzes nicht verändert werden, weil ihr höchster Wert nicht einen Datensatz, sondern den Klammeraffen »@« gezählt hat. Der aber soll durch den neuen Datensatz überschrieben werden.

Erst wenn in Zeile 470 die Frage nach einer weiteren Eingabe mit J(a) beantwortet wird, muß D um 1 erhöht werden. Wenn nach der Frage kein J(a) erfolgt, springt die Zeile 480 auf das Unterprogramm »DATEI ABSPEICHERN«, damit die Eingabe nicht verlorengeht.

Dieses Unterprogramm speichert die nunmehr vergrößerte Datei natürlich wieder unter dem Namen »LITERATUR« ab. Eine Datei mit diesem Namen existiert aber schon auf der Diskette oder der Kassette. Deshalb muß bei Diskettenbetrieb in Zeile 510 die sogenannte »Save and Replace«-Funktion verwendet werden:

```
→ 510 OPEN 1,8,4,"@:LITERATUR,S,W"
```

Das Einfügen des Klammeraffen »@« gefolgt von einem Doppelpunkt bewirkt, daß die neue Datei die alte Datei desselben Namens überschreibt.

Bei Kassettenbetrieb wird die neue, vergrößerte Datei einfach neu geSAVEt.

Und noch etwas fehlt: die Endmarkierung, die wir in Zeile 150 abfragen und die durch die neue Eintragung überschrieben worden ist.

Sie muß am Ende des Unterprogramms, noch vor dem CLOSE-Befehl, per PRINT#-Befehl an die Datei angehängt werden:

```
→ 560 PRINT#1,"@"
```

Jetzt kommt nur noch das Unterprogramm »ENDE« ab Zeile 600 dazu, das aber denkbar einfach ist:

```
600 REM--------- ENDE ---------
610 END
```

Das komplette Programm ist in Listing 10 wiedergegeben.

LEKTION 31: Die Booleschen Funktionen AND, OR, NOT

In Lektion 8 bei den Prüfungen habe ich als Prüfbedingungen neben den Vergleichsfunktionen auch die sogenannten Boole'schen Funktionen

AND, OR und NOT

erwähnt, ohne aber näher darauf einzugehen. Das will ich hier nachholen. Sie sind prinzipiell bei zwei Verfahren einsetzbar.

31.1. Verwendung als Prüfbedingung

Die Wirkungsweise kann am besten an einem Beispiel demonstriert werden. Ich nehme dazu aus Lektion 8 den vierten Anwendungsfall, bei dem wir zwei IF-THEN-Befehle hintereinander gesetzt haben:

```
350 IF X=4 THEN IF Y=0 THEN PRINT "JA"
```

Diese Zeile kann auch mit AND geschrieben werden:

```
350 IF X=4 AND Y=0 THEN PRINT "JA"
```

Die Wirkungsweise von AND entspricht seiner Übersetzung "UND". Erst wenn beide Bedingungen (X=4 und Y=0) erfüllt sind, wird die Aktion hinter dem THEN-Befehlsteil ausgeführt.

Die OR-Funktion wirkt entgegengesetzt:

```
350 IF X=4 OR Y=0 THEN PRINT "JA"
```

Jetzt wird die Aktion nach THEN dann ausgeführt, wenn entweder die eine Bedingung (X=4) oder die andere (Y=0) oder beide erfüllt sind. Um das auszuprobieren, nehmen wir die gesamte verknüpfte Schleife, aus der die Zeile 350 stammt.

```
310 X=2:Y=1
320 X=X+Y-2
```

```
2 REM++++++++ LITERATUR DATEI +++++++++    <196>
3 :                                        <235>
4 :                                        <236>
5 REM--------- MENUE ----------------      <158>
7 :                                        <239>
10 PRINT CHR$(147)                         <039>
15 PRINT SPC(90)"WAEHLEN SIE BITTE AUS"     <223>
20 PRINT TAB(202)"(1) DATEI LADEN"         <047>
25 PRINT SPC(82) "(2) DATENSAETZE SUCHEN"  <019>
30 PRINT SPC(82) "(3) NEUEN TEXT EINGEBEN" <253>
35 PRINT SPC(82) "(4) DATEI ABSPEICHERN"   <003>
40 PRINT SPC(82) "(5) ENDE"                <021>
45 :                                       <021>
50 PRINT                                   <152>
55 INPUT A                                 <087>
60 ON A GOSUB 100,200,400,500,600          <110>
65 GOTO 10                                 <243>
90 :                                       <066>
100 REM-------- DATEI LADEN ----------     <044>
105 :                                      <081>
110 DIM A$(100,6)                          <230>
115 OPEN 1,8,3,"LITERATUR,S,R"             <250>
120 D=D+1                                  <003>
130 FOR K=1 TO 6                           <119>
140 INPUT#1,A$(D,K)                        <064>
150 IF A$(D,K)="@"THEN K=6:NEXT K:GOTO 180 <049>
160 NEXT K                                 <004>
170 GOTO 120                               <130>
180 CLOSE 1                                <191>
190 RETURN                                 <248>
195 :                                      <171>
200 REM------ DATENSAETZE SUCHEN ------    <093>
205 :                                      <181>
210 INPUT"STICHWORT";S$                    <148>
220 FOR S=1 TO D                           <024>
230 FOR K=1 TO 6                           <219>
240 IF A$(S,K)=S$ THEN FOR Z=1 TO 6:PRINT
    A$(S,Z):NEXT Z                         <162>
250 NEXT K                                 <094>
260 PRINT                                  <108>
270 NEXT S                                 <180>
280 PRINT"ENDE DER SUCHE"                  <165>
290 PRINT                                  <138>
300 :                                      <022>
310 PRINT"NOCH EINMAL (J/N)?"              <064>
320 GET V$:IF V$="" THEN 320               <182>
330 IF V$="J" THEN 210                     <054>
340 RETURN                                 <144>
390 :                                      <112>
400 REM----- NEUEN TEXT EINGEBEN ------    <029>
405 :                                      <127>
420 K=1                                    <123>
430 PRINT K".KATEGORIE":INPUT A$(D,K)      <060>
440 IF K<6 THEN K=K+1:GOTO 430             <246>
445 :                                      <167>
450 PRINT:PRINT"NOCH EINE EINGABE (J/N) ?" <231>
460 GET V$:IF V$="" THEN 460               <099>
470 IF V$="J" THEN D=D+1:GOTO 420          <080>
480 GOTO 500                               <186>
490 :                                      <212>
500 REM------- DATEI ABSPEICHERN ------    <220>
505 :                                      <227>
510 OPEN 1,8,4,"@:LITERATUR,S,W"           <217>
520 FOR S=1 TO D                           <072>
530 FOR K=1 TO 6                           <011>
540 PRINT#1,A$(S,K)                        <051>
550 NEXT K:NEXT S                          <085>
560 PRINT#1,"@"                            <070>
570 CLOSE 1                                <073>
580 RETURN                                 <130>
599 :                                      <067>
600 REM----------- ENDE --------------     <045>
605 :                                      <073>
610 END                                    <104>
```


Listing 10. Die Literatur-Datei


```
330 Y=Y-X+3
340 PRINT X;Y
350 IF X=4 AND Y=0 THEN PRINT "JA"
360 GOTO 320
```

Das Resultat ist wie gesagt das gleiche wie in Lektion 8. Die Schleife druckt das JA nach dem fünften Umlauf aus.

Wenn Sie in Zeile 350 die OR-Funktion verwenden, wird das JA schon nach dem dritten Umlauf und dann noch einmal nach dem fünften Umlauf ausgedruckt.

Ich will Ihnen noch ein Beispiel zeigen:

```
350 IF (X AND Y)=4 THEN PRINT "JA"
```

Diese Bedingung wird offensichtlich überhaupt nicht erfüllt, denn das JA erscheint nicht. Steht dagegen in der Klammer

```
350 IF (X OR Y)=4 THEN PRINT "JA"
```

dann erscheint das JA und zwar nach dem fünften Umlauf bei Kombination X=4, Y=0.

Sie sehen also, daß X AND Y nicht gleichbedeutend ist mit X + Y. Das Resultat der beiden Bedingungen ist auf Anhieb nicht verständlich. Das liegt daran, daß diese beiden Funktionen die Variablenwerte im Dualsystem und da auch noch »Bit-weise« verarbeiten und zwar nach folgenden Regeln:

UND	ODER
0 AND 0 = 0	0 OR 0 = 0
0 AND 1 = 0	0 OR 1 = 1
1 AND 0 = 0	1 OR 0 = 1
1 AND 1 = 1	1 OR 1 = 1

Ich gehe hier davon aus, daß Sie, das duale (oder auch binäre) Zahlensystem, das nur aus den Ziffern 0 und 1 aufgebaut ist, kennen. Ich habe nämlich hier keine Gelegenheit, darauf einzugehen. Wenn Sie es nicht kennen, dann bitte ich Sie in einem entsprechenden Computerbuch nachzulesen.

Nach obigen Rechenregeln ergibt die Aufgabe

4 AND 3 ergibt 0
4 OR 3 ergibt 7

Im Detail gerechnet sieht das nämlich so aus:

4:0000 0100
3:0000 0011

AND:0000 0000 (entspricht 0)
DOR:0000 0111 (entspricht 7)

Jetzt können wir aus den Läufen der Zeilen 310 bis 350 eine Ergebnistabelle machen, die wir auch verstehen können.

	X	Y	X=4 AND Y=0	X=4 OR Y=0	(X AND Y)=4	(X OR Y)=4
Anfang	2	1	nein	nein	nein (0)	nein (3)
1.Lauf	1	3	nein	nein	nein (1)	nein (3)
2.Lauf	2	4	nein	nein	nein (0)	nein (6)
3.Lauf	4	3	nein	JA	nein (0)	nein (7)
4.Lauf	5	1	nein	nein	nein (1)	nein (5)
5.Lauf	4	0	JA	JA	nein (0)	JA (4)
6.Lauf	2	1	nein	nein	nein (0)	nein (3)
und so weiter						

Ganz rechts in der Tabelle stehen die nach den obigen Regeln sich ergebenden Werte in Klammern.

Die dritte Boolesche Funktion

NOT

wird nur sehr selten verwendet. Auch sie verarbeitet die Werte bit-weise im Dualsystem und zwar vertauscht sie einfach jede 1 mit einer 0 und umgekehrt jede 0 mit einer 1.

5: 0 0000 0101
NOT 5: 1 1111 1010

Diese resultierende Dualzahl 1111 1010 mit noch einer 1 davor entspricht in der speziellen Rechenweise des C64 der negativen Zahl –6.

Auch auf dieses Phänomen – es entsteht durch die spezielle Darstellung der negativen Zahlen im Dualsystem – gehe ich hier nicht näher ein, sondern gebe nur die Formel für NOT an:

NOT X bedeutet -(X+1)

Sie sehen, eine solche Funktion läßt sich nicht ohne vorbereitendes Kopfrechnen anwenden. Bei der Formel für NOT ist die Klammer sehr wichtig, bei den beiden rechten Bedingungen in der letzten Tabelle auch:

IF (X AND Y)=4 ist nicht dasselbe wie

IF X AND Y=4

Probieren Sie es mit der verschachtelten Schleife aus!

31.2. Die Klammer-Regeln

Wir müssen eine weitere Regel beachten, allerdings nicht nur bei den Boole'schen Funktionen, sondern auch bei den arithmetischen Funktionen. Anlaß zu dieser Regel bietet die Möglichkeit, in einer IF-Prüfung mehrere Funktionen abzufragen. Es ist, in Erweiterung zu dem was wir ja schon ausprobiert haben, durchaus zulässig, die folgende Bedingung zu machen:

IF A=1 AND B=2 OR C=3 THEN...

Nur muß man beachten, daß diese Zeile so wirkt, als wäre sie mit den folgenden Klammern geschrieben:

IF (A=1 AND B=2) OR C=3 THEN...

Es wird also zuerst die AND-Verknüpfung gebildet und erst dann die OR-Verknüpfung mit dem Resultat der Klammer.

Dieselbe Zeile, wie folgt geschrieben, bringt ein anderes Resultat:

IF A=1 AND (B=2 OR C=3) THEN...

Hier folgt die AND-Verknüpfung erst nach der OR-Verknüpfung. Die Klammern sind deswegen notwendig, weil die einzelnen Funktionen eine eigene »Hackordnung« haben, das heißt, sie haben verschiedene Prioritäten, die in der folgenden Tabelle dargestell sind.

1. ↑ Potenz
2. Vorzeichenwechsel
3. * / Multiplikation, Division (gleichberechtigt)
4. + – Addition, Subtraktion (gleichberechtigt)
5. < > = Vergleichsfunktionen (gleichberechtigt)
6. AND
7. NOT
8. OR

Um Schwierigkeiten zu vermeiden, ist es empfehlenswert, im Zweifelsfall immer Klammern einzusetzen.

31.3. Verwendung als Bit-Maske

In Lektion 21 haben wir beim Stöbern im Speicher gesehen, daß es Speicherzellen gibt, deren Inhalt die Arbeitsweise des Computers beeinflussen. Derartige Speicherzellen nennen wir REGISTER.

So schaltet zum Beispiel die Zahl 5 in das Register 53820 gePOKEt die Farbe des Bildschirmspeichers auf Grün. In den meisten Registern des C64 übt ein einzelnes Bit oder Bitgruppen eine eigene Steuerfunktion aus. Als Beispiel führe ich das Register 53265 an:

Bit-Nr.	7	6	5	4	3	2	1	0
Stellenwert	128	64	32	16	8	4	2	1

Bit 0–2: bestimmt den Wert der Zeilenverschiebung in Y-Richtung beim Smooth-Scrolling

Bit 3: schaltet Zeilenzahl (0=24 Zeilen, 1=25 Zeilen)

Bit 4: schaltet den Bildschirm (0=gelöscht)

Bit 5: schaltet Hochauflösungsmodus (1=eingeschaltet)

Bit 6: schaltet veränderten Hintergrund-Farbmodus (1=eingeschaltet)

Bit 7: letztes Bit des Farbregisters 53266

Diese einzelnen Funktionen des Registers 53265 werden Ihnen jetzt nicht allzuviel sagen. Wenn Sie mehr über dieses und noch andere Register wissen und lernen wollen, dann verweise ich Sie auf den Aufsatz von Thomas Erhart »Der leichte Umgang mit Sprites« im 64'er Sonderheft 5/86 ab Seite 90. Ich will Ihnen aber zeigen, wie Sie mit Hilfe der Boole'schen Funktionen einzelne Bits eines Registers setzen (auf 1) oder löschen (auf 0) können, ohne die anderen Bits zu beeinflussen. An sich ist es ganz einfach. Man nimmt nämlich nur den gegenwärtigen Inhalt des Registers – mit dem PEEK-Befehl – verknüpft ihn über die AND- oder OR-Funktion mit einer MASKE und POKEt das ganze wieder zurück in das Register.

Was ist eine Maske und wie sieht sie aus?

Bit setzen:

Wenn wir zum Beispiel das 4. Bit des Registers – es hat den Stellenwert 16 – auf 1 setzen wollen, dann hilft uns die OR-Funktion als Maske.

```
PEEK (53265)      x x x x   x x x x
Maske für 4. Bit  0 0 0 1   0 0 0 0
-----------------------------------
              OR  x x x 1   x x x x
```

An der Stelle, wo in der Maske eine 1 steht, wird laut Rechenregel (siehe Abschnitt 31.1) durch OR das entsprechende Bit auf 1 gesetzt, alle anderen Bits behalten ihren ursprünglichen Wert.

Der Befehl dazu lautet:

```
POKE 53265,PEEK (53265) OR 16
```

Die Maske zum Setzen des 0. Bit und des 7. Bit schaut demnach so aus:

```
POKE 53265,PEEK (53265) OR (128+1)
```

Bit löschen:

Die Methode ist dieselbe, nur verwenden wir jetzt die AND-Funktion als Maske.

```
PEEK (53265)      x x x x   x x x x
Maske für 4. Bit  1 1 1 0   1 1 1 1
-----------------------------------
             AND  x x x 0   x x x x
```

Überall dort, wo in der Maske eine 1 steht, bleibt laut Rechenregel mit AND der ursprüngliche Wert erhalten, nur das Bit mit einer 0 in der Maske wird ebenfalls auf 0 »gelöscht«.

Zur Berechnung der Zahl für die Maske wird einfach der Stellenwert des zu löschenden Bit von 255 abgezogen.

Um Bit 4 zu löschen, lautet der Befehl:

```
POKE 53265,PEEK (53265) AND (255-16)
```

LEKTION 32: Die arithmetischen und geometrischen Funktionen

Diese Funktionen, hinter denen immer in Klammern ein sogenanntes Argument steht, führen mit dem Argument eine bestimmte Berechnung durch.

Alle derartige Funktionen des C64 finden Sie auch auf einem Schultaschenrechner. Ich werde deshalb – Ihr Wissen voraussetzend – hier nur eine Übersicht geben.

SIN (X)

- berechnet den Sinus von X, wobei das Argument X im Bogenmaß angegeben wird
- wird das Argument in Grad angegeben, so muß folgende Formel verwendet werden:

SIN (X*π/180)

π erzeugen Sie durch die Tastenkombination <SHIFT> und <↑>

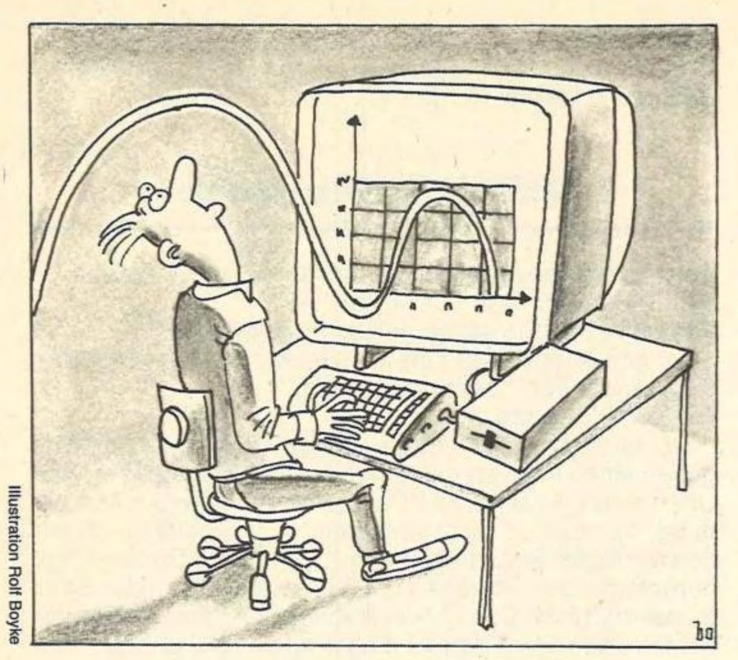


Illustration Rolf Boyke

Das folgende Beispiel druckt eine senkrechte Sinuskurve aus:

```
10 PRINT CHR$(147)
20 FOR X=0 TO 6 STEP 0.29
30 A=INT(20+18*SIN(X))
40 PRINT TAB(A);"*"
50 NEXT X
60 END
```

COS (X)
- berechnet den Kosinus von X, wobei das Argument X im Bogenmaß angegeben wird
- wird das Argument in Grad angegeben, so muß folgende Formel verwendet werden:

COS (X*π/180)

Das folgende Beispiel druckt eine senkrechte Kosinuskurve aus:

```
110 PRINT CHR$(147)
120 FOR X=0 TO 6 STEP 0.29
130 A=INT(20+18*COS(X))
140 PRINT TAB(A);"*"
150 NEXT X
160 END
```

TAN (X)
- berechnet den Tangens von X, wobei das Argument X im Bogenmaß angegeben wird
- wird das Argument in Grad angegeben, so muß folgende Formel verwendet werden:

TAN (X*π/180)

ATN (X)
- berechnet den Arcustangens von X, wobei das Argument X im Bogenmaß angegeben wird
- wird das Argument in Grad angegeben, so muß folgende Formel verwendet werden:

ATN (X*π/180)

ABS (X)
- liefert den Absolutwert des Arguments X, das heißt, positive Werte bleiben unverändert, negative Werte werden in positive Werte umgewandelt

Im folgenden Beispiel wird geprüft, ob eine eingegebene Zahl positiv oder negativ ist durch Division der Zahl mit ihrem absoluten Wert.

```
210 INPUT X
220 IF X/ABS(X) =-1 THEN PRINT "NEGATIV"
230 IF X/ABS(X) =1 THEN PRINT "POSITIV"
240 END
```

EXP (X)
- liefert den »natürlichen Exponenten« der Zahl X, oder in Worten ausgedrückt:

e hoch X

(e=2.71828183.. und ist die Basis des natürlichen Logarithmus)
- EXP ist die Umkehrfunktion von LOG.

LOG (X)
- liefert den »natürlichen Logarithmus« (mit Basis e) der Zahl X, die immer positiv sein muß
- LOG ist die Umkehrfunktion von EXP.

SGN (X)
- liefert das Vorzeichen des Arguments X.
- die Funktion kann nur die Werte 1, 0 oder -1 annehmen nach folgender Regel:

bei X = 0 SGN(X)=0
bei X < 0 SGN(X)=-1
bei X > 0 SGN(X)=1

SQR (X)
- berechnet die Quadratwurzel der Zahl X. X muß immer positiv sein
- SQR ist die Umkehrfunktion zum Potenzieren mit ↑2

Im Grunde genommen gehören die Funktinen INT und RND auch in diese Gruppe. Wir haben sie aber bereits in Lektion 10 »Zufalls- und ganze Zahlen« im Detail behandelt.

LEKTION 33: Selbstdefinierte Funktionen

Wem die bisher vorgestellten Funktionen, die uns Basic bietet, nicht genügen, dem erlaubt Basic seine eigenen Funktionen zu definieren.

Daß ein derartiger Wunsch durchaus vorkommen kann, will ich Ihnen an einigen Beispielen zeigen.

(1) Die Fläche F eines Kreises wird berechnet mit

F=R*R*π

(2) Die Oberfläche O einer Kugel wird berechnet mit

O=R*R*π*4

(3) Das Volumen eines Kegels mit kreisförmiger Grundfläche wird berechnet mit

V=R*R*π*h/4

Der Ausdruck R*R*Pi kommt also recht häufig vor. Wenn Sie jetzt die Aufgabe hätten, für 50 verschiedene Werte des Radius R – R(1) bis R(50) – die entsprechenden Flächen F(1) bis F(50) auszurechnen, dann müßten Sie den Ausdruck R*R*π recht oft verwenden.

Hier lohnt es sich also, dafür eine eigene Funktion zu »erfinden« und ihr einen Namen zu geben, dem man dann wie einer Variablen die verschiedenen Werte zuordnen kann.

Dazu gibt es in Basic den Befehl

DEF FN

(abgeleitet aus DEFine FunktioN).

In unserem obigen Beispiel können wir eine Funktion definieren, der wir den Namen RRPI geben wollen:

DEF FN RRPI(R) = R*R*π

In allen obigen Formeln können wir nun schreiben:

F=FN RRPI(R)
O=FN RRPI(R)*4
V=FN RRPI(R)*h/4

Der eigentliche Pfiff bei dieser selbstdefinierten Funktion liegt in der Eigenschaft der Variablen in der Klammer, in unserem Fall ist es das R. Diese Variable ist nämlich »übertragbar«. Was heißt das?

Wenn wir unsere selbstdefinierte Funktion FN RRPI so verwenden:

X=23+FN RRPI(R)-3

dann lautet die Funktion: R*R*π

Schreiben wir aber:
X=23+FN RRPI(K1)-3
dann lautet die Funktion: K1 * K1 * π

Die Zuordnung geschieht innerhalb des DEF-Befehls. Leider ist es nicht möglich, mehr als eine deartige übertragbare Variable zu verwenden. Schauen Sie sich das folgende Beispiel an:

```
10 X=20
20 DEF FN Z(R) = 10*INT(2*R+X)
30 X=0.5
40 PRINT FN Z(3.1)
```

In Zeile 20 haben wir in der Formel 2 Variable, R und X. Aber nur R ist als Argument in der Funktion FN Z enthalten. Ihr weisen wir in Zeile 40 den Wert 3.1 zu.

X ist die andere Variable, die vor der Definition der Zeile 20 auf den Wert 20 gesetzt wird. Nach der Definition erhält sie aber den Wert 0.5 (Zeile 30). Und nur das ist der Wert, der in der Berechnung der Zeile 40 verwendet wird:

```
10*INT(2*3.1+0.5) = 10*INT(6.7)
                  = 10*6
                  = 60
```

Als Einschränkung gilt übrigens, daß man mit DEF FN nur numerische Funktionen definieren darf. Strings haben da keinen Platz.

Basic-Befehl Nr. 41 DEF FN
- der Befehl wird so geschrieben:
  DEF FN Name(Variable) = Formel
- mit diesem Befehl können eigene komplizierte numerische Funktionen mit einem kurzen Namen definiert werden
- der Name der Funktion beginnt immer mit FN
- der jeweilige Wert der in Klammern stehenden numerischen Variablen wird überall in der Formel eingesetzt, wo die Variable steht
- die Funktion wird nach der Definition aufgerufen mit:
  Basic-Befehl FN Name ( )
- DEF FN darf nicht im Direktmodus verwendet werden, String-Variable sind nicht zugelassen

Neben den wenigen mathematischen Funktionen, die in Lektion 33 aufgeführt sind, gibt es viele andere, die so wichtig sein können, daß Commodore im Handbuch für den C 64 auf Seite 140 eine eigene Tabelle dafür angelegt hat.

Diese Funktionen, die in Basic nicht vorhanden sind, eignen sich natürlich ideal für die Selbstdefinition:

Arcussinus:
DEF FN ARCSIN(X)=ATN(X/SQR(1-X↑2))

Sinus Hyperbolikus:
DEF FN SINH(X)=(EXP(X)-EXP(-X))/2

Interesant ist übrigens, daß es uns durchaus erlaubt ist, eine selbstdefinierte Funktion als Variable (Argument) einer zweiten selbstdefinierten Funktion zu verwenden.

Als Beispiel wähle ich die Formel für das Volumen einer Kugel, deren Ergebnisse ich mit einer anderen Funktion auf 2 Dezimalstellen aufrunde.

Funktion für das Kugelvolumen
```
10 DEF FNV(R)=4*π*R↑3/3
```
Funktion zum Aufrunden
```
20 DEF FNA(X)=INT((X+0.005)*100)/100
```
Der Rest des Programms lautet:
```
30 INPUT "RADIUS";R
40 PRINT FNV(R);
50 PRINT FNA(FNV(R))
60 GOTO 30
```

Mit diesen kleinen Programm wird in Zeile 40 zuerst das dem eingegebenen Radius entsprechende Volumen der Kugel ausgedruckt und mit dem Semikolon danebengesetzt, danach durch Zeile 50 der aufggerundete Wert. Dabei ist einfach die Volumenfunktion FNV an die Stelle der Variable X gesetzt worden.

LEKTION 34: Fehlersuche

Beim Programmieren gilt ein beinahe ehernes Gesetz: Kein Programm läuft auf Anhieb!
Wie oft haben Sie schon geflucht und gesucht?

Es gibt aber einige Tips und Tricks, wie man sich das Suchen der Fehler erleichtern kann.

Fehlermeldungen

Die häufigsten Fehler treten durch Tippfehler beim Eingeben eines Programms auf. In vielen Fällen reagiert das Übersetzungssystem des Computers mit einer Fehlermeldung, die aber oft nicht verständlich ist. Ehe Sie sich mit den dürftigen Erklärungen des Commodore-Handbuches herumschlagen, ist es besser, Sie besorgen sich das 64'er Sonderheft 5/86. Dort hat der Redakteur Achim Hübner alle Fehlermeldungen zusammengetragen und anhand von guten Beispielen erklärt. Es lohnt sich schon, die Bedeutung der Fehlermeldungen zu kennen.

Programmunterbrechung

Wenn ein Programm etwas anderes tut, als Sie wollen, ist noch lange nicht klar, wo der Fehler liegt. Um das herauszufinden, ist es empfehlenswert, nur Teile des Programms laufen zu lassen. Das geht aber meistens nur vom Anfang an – also halten Sie am besten das Programm an ganz bestimmten Stellen an. Wenn es bis dahin nichts falsch gemacht hat, dann haben Sie den Bereich, in dem der Fehler liegen muß, schon beachtlich eingeschränkt.

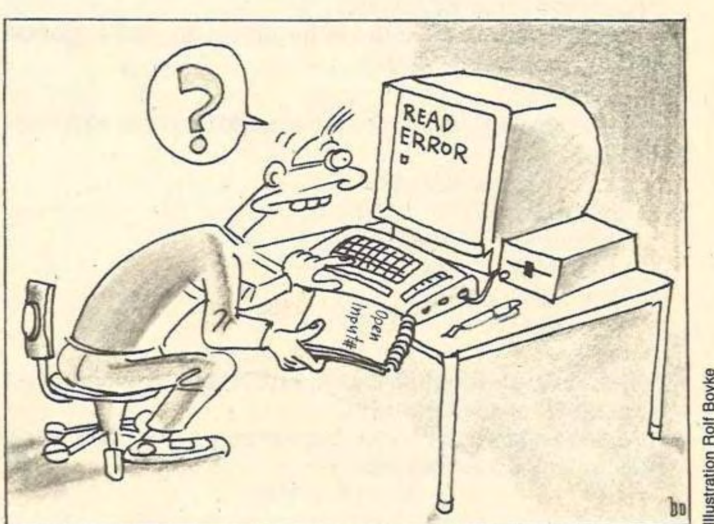


Illustration Rolf Boyke

Zum Anhalten eines Programms gibt es bekanntlich den Befehl END, durch den das Programm abbricht und der Computer sich mit READY meldet.
Es gibt aber noch einen zweiten derartigen Befehl:
STOP
der wie END auch das Programm abbricht, aber mit der Meldung
BREAK IN xxx
die Zeile xxx angibt, wo der STOP-Befehl steht.

Durch diesen Stopp werden die Werte der Variablen nicht verändert und sie können mit einem Direktbefehl ausgedruckt und überprüft werden. Ich komme gleich darauf noch zurück.

Wichtig ist es, daß das gestoppte Programm in der nach dem STOP kommenden Zeile fortgesetzt werden kann, mit

GOTO yyy, wenn man die Zeilennummer yyy kennt, oder aber mit dem neuen Basic-Befehl

CONT

was aus dem englischen Wort *Continue* abgeleitet ist. Voraussetzung für das Funktionieren dieses Befehls ist allerdings, daß vorher keine Änderungen im Programm vorgenommen worden sind.

Wenn Sie aber nach dem STOP geLISTet und einen Fehler korrigiert haben, müssen Sie mit RUN wieder von vorn anfangen.

Im Extremfall können Sie in jede Programmzeile ein STOP einbauen und mit CONT zeilenweise das Programm durchgehen, bis Sie zu der kritischen Stelle kommen.

Am Ende, wenn alles läuft, müssen Sie natürlich alle STOPs wieder entfernen.

Variable ausdrucken

Eine Fehleranalyse ist oft dadurch möglich, daß man sieht, welche Werte die verschiedenen Variablen im Lauf eines Programms annehmen.

Einige zusätzliche PRINT-Befehle, zum Beispiel innerhalb einer Schleife, verschandeln zwar den gewünschten Ausdruck auf dem Bildschirm, geben aber Auskunft und Einsicht über den Ablauf und was da so passiert.

Wie oben schon erwähnt, kann man nach einem STOP und ohne ein späteres CONT zu stören, mit PRINT im Direktmodus (also ohne Zeilennummern) den aktuellen Wert einer oder mehrerer Variablen ausdrucken.

Eine etwas seltenere Art der Variablenbehandlung bietet uns ein weiterer Basic-Befehl

CLR

der von CLEAR, also Löschen abgeleitet ist.

Vorsicht! der Befehl CLR hat nichts mit der CLR-Taste gemein.

CLR bezieht sich in seiner »Lösch«-Funktion nur auf den Speicher, während die CLR-Taste den Bildschirm löscht. Mit dem CLR-Befehl werden alle numerischen Variablenwerte auf 0 gesetzt, allen Stringvariablen wird ein »Nullstring« zugewiesen und alle Dimensionen von Feldern werden gelöscht. Wenn DATA-Zeilen vorhanden sind, die ganz oder teilweise gelesen wurden, wird die Funktion von RESTORE ausgeführt. Schließlich werden im RAM-Speicher des Betriebssystems alle vom Programm eingestellten Werte auf ihren Zustand nach dem Einschalten des Computers zurückgesetzt.

Nur zwei Dinge bleiben unberührt - und das ist sehr wichtig: das im Speicher befindliche Programm und Datei-Befehle (OPEN).

Die Funktion von CLR wird übrigens auch bei jedem RUN-, LOAD- und NEW-Befehl ausgeführt.

Der CLR-Befehl kann sehr nützlich sein, um sozusagen die Ausgangslage eines Programms innerhalb eines Programm-Ablaufes wieder herzustellen.

Basic-Befehl Nr. 42 STOP
- dieser Befehl bricht ein Programm ab
- im Unterschied zum END-Befehl meldet sich der Computer nach dem STOP-Befehl mit der Meldung BREAK IN xxx, wobei mit xxx die Zeilennummer angegeben wird, in der das Programm abgebrochen worden ist

Basic-Befehl Nr. 43 CONT
- mit diesem Direktbefehl kann man ein gestopptes Programm, an dem nach dem STOP keine Veränderungen vorgenommen sein dürfen, ab der STOP-Stelle weiterlaufen lassen. Die entsprechende Zeilennummer braucht nicht angegeben zu werden. Alle Werte der Variablen bleiben erhalten

Basic-Befehl Nr. 44 CLR
- dieser Befehl löscht die Werte aller Variablen und Dimensionen von Feldern. Numerische Variable erhalten den Wert 0, String-Variable einen Nullstring. Der Zeiger von READ-DATA-Befehlen wird auf die erste DATA-Zeile gesetzt.
- der Zustand von eröffneten Dateien und ein im Speicher stehendes Programm bleiben unberührt.

LEKTION 35: Seltene Basic-Befehle

In dieser letzten Lektion will ich die Befehle

WAIT, SYS, USR

und die reservierte Status-Variable

ST

behandeln.

Diese vier Kandidaten sind nicht umsonst selten anzutreffen, verlangen doch alle außer WAIT ein fortgeschrittenes Wissen der Programmierung.

Ich bin mir klar darüber, daß für einen derartigen Einführungskurs in Basic SYS, USR und ST zu schwer sind. Ich bringe aber dieses Kapitel der Vollständigkeit halber und entschuldige mich gleich, daß ich es nicht für Einsteiger schreiben kann.

53.1. Der WAIT-Befehl

Normalerweise, wenn in einem Programm gewartet werden soll, verwenden wir eine GET-Schleife vom Typ:

```
10 GET A$:IF A$="" THEN 10
```

In Basic gibt es aber einen - viel zu selten verwendeten - Befehl, der die Programmausführung solange unterbricht, bis in einer angegebenen Adresse des Speichers etwas verändert wird. Er lautet:

WAIT

Hinter dem WAIT-Befehl, der übersetzt Warten heißt, muß die Adresse einer beliebigen Speicherzelle stehen, in der etwas passieren soll. Zusätzlich folgen auf die Adresse, durch Komma getrennt, eine oder zwei numerische Variable. Was das alles soll, zeigt am besten ein Beispiel.

In Lektion 9 »Eingabe« habe ich den Tastaturpuffer beschrieben, in den Zahlen und Zeichen abgespeichert werden, wenn ihre Tasten während des Programmlaufs gedrückt werden.

In der Speicherzelle 198 nun steht immer die aktuelle Zahl der im Tastaturpuffer gespeicherten Zeichen.

```
100 POKE 198,0
```

Mit dieser Zeile wird »künstlich« der Puffer geleert.

```
110 WAIT 198,1
```

Dieser Befehl unterbricht das Programm solange, bis irgendeine Taste gedrückt worden ist, wodurch natürlich in Speicherzelle 198 eine 1 steht.

Ein anderes Beispiel bietet die Speicherzelle 653. Sie enthält eine Zahl, die angibt, welche der Steuertasten SHIFT, CTRL oder CBM gedrückt worden ist und zwar:

SHIFT = 1
CBM = 2
CTRL = 4

Auch diese Tasten können wir mit WAIT abfragen:

```
200 PRINT "A"
210 WAIT 653,4
230 GOTO 200
```

Mit diesen drei Zeilen wird zuerst ein A ausgedruckt, dann wartet Zeile 210 auf die CTRL-Taste und druckt, solange sie gedrückt ist, lauter A auf den Bildschirm. Um zu erreichen, daß pro Tastendruck nur ein einziges A ausgedruckt wird, fügen wir Zeile 220 ein:

```
220 WAIT 653,4,4
```

Was da passiert, bedarf einer nicht sehr leicht verständlichen Erklärung, die in der Befehlsbeschreibung steht.

Basic-Befehl Nr. 45 WAIT
- Er wird so geschrieben:
  WAIT Adresse,N1,N2
- Er unterbricht ein Programm solange, bis der Inhalt der mit der Adresse angegebenen Speicherzelle von Null verschieden ist.
- Die Veränderung des Inhalts der adressierten Speicherzelle wird durch die beiden numerischen Variablen N1 und N2 erzeugt, und zwar dadurch, daß zuerst der Inhalt der Speicherzelle mit N2 über ein EXOR verknüpft wird. Dieses Ergebnis wird dann mit N1 über ein AND verknüpft.
  Ist das Resultat ungleich 0, bleibt das Programm unterbrochen.
  Ist das Resultat gleich 0, fährt das Programm mit dem nächsten Befehl fort.
- N2 kann auch weggelassen werden. Dann wird es automatisch auf 0 gesetzt.

Um diese Berechnung zu verstehen, muß ich zuerst die Boole'sche Funktion EXOR erklären, die leider im C64 nicht für den generellen Gebrauch eingebaut ist.

Ähnlich wie das in Lektion 31 beschriebene OR gilt für das XOR folgende Verknüpfungsregel:

0 XOR 0 = 0
0 XOR 1 = 1
1 XOR 0 = 1
1 XOR 1 = 0

Betrachten Sie bitte nochmal die Zeilen 200 bis 230.

```
210 WAIT 653,4
```

```
Inhalt 653   = 0    0000
N2           = 0    0000
XOR                 ____
                    0000
N1           = 0    0000
AND                 ____
Ergebnis            0000
```

das heißt, das Programm wartet.

Wenn die CTRL-Taste gedrückt wird, steht in 653 eine 4. Die obige Rechnung ergibt 4 und das Programm läuft weiter.

Dieselbe Rechnung für Zeile 220 sieht so aus:

```
220 Wait 653,4,4
```

```
Inhalt 653   = 4    0100
N2           = 4    0100
XOR                 ____
                    0000
N1           = 4    0100
AND                 ____
Ergebnis            0000
```

das heißt, das Programm wartet wieder.

35.2. Der SYS-Befehl

Sowohl eigene Programme, die in Maschinensprache geschrieben sind und daher im RAM-Speicher stehen, als auch Programmteile (Routinen) des Betriebssystems und des Basic-Übersetzers, die im ROM-Speicher stehen, können von einem Basic-Programm aus gestartet (aufgerufen) werden. Dieser Befehl entspricht damit dem RUN-Befehl für Basic-Programme. Er wird so geschrieben:

SYS Adresse

Wichtig ist, daß am Ende der mit SYS gestarteten Maschinensprache-Routine ein RTS-Befehl steht, durch den die Programmkontrolle wieder an das Basic-Programm zurückgegeben wird. Ein Beispiel dafür ist

SYS 58260

der Sprung auf den Kaltstart, der den Computer in die Ausgangslage zurücksetzt.

Die meisten dieser Routinen benötigen jedoch verschiedene Angaben – man nennt sie auch Parameter – die vor der Ausführung des SYS-Befehls richtig eingestellt sein müssen.

Die LOAD-Routine zum Beispiel, die ab Speicherzelle 62622, beginnt, können wir mit

SYS 62622

nicht starten. Es fehlen die Angaben über Geräte-Nummer (8 für Floppy, 0 für Band), File-Namen, sowie Anfangs- und Endadresse. Diese Parameter werden normalerweise nach dem Befehl LOAD von der Routine des Interpreters, die den LOAD-Befehl übersetzt, eingegeben. Wir geben ja nicht einfach LOAD ein, wenn wir ein Programm mit dem Namen »TEST« auf Diskette speichern wollen, sondern wir schreiben

```
LOAD "TEST",8
```

Auch wenn wir nur LOAD eintippen, werden vom Übersetzter Parameter gesetzt, nämlich »namenlos« und 0 für Bandgerät. Ich hoffe, Ihnen ist geläufig, daß bei Weglassen aller Angaben der Übersetzter immer Kassettenoperationen durchführt.

Natürlich können wir uns das anschauen: Die Routine des Übersetzters für den BASIC-Befehl LOAD beginnt an Speicherzelle 57704 .

Mit SYS 57704 springen wir dorthin – und in der Tat, wir erhalten auf dem Bildschirm PRESS PLAY ON TAPE. Aber ein Programm auf diese Weise auf die Floppy zu LOADen, gelingt uns nicht, es sei denn, wir können die fehlenden Parameter von Hand eingeben.

Genau das aber können wir, weil der SYS-Befehl sich diese Parameter aus den Speicherzellen 780 bis 783 holt und in die vier Register des Mikroprozessors schreibt.

780 ist die Adresse des Akkumulators
781 ist die Adresse des X-Registers
782 ist die Adresse des Y-Registers
783 ist die Adresse des P-Registers.

Die Behandlung von A, X und Y ist unkompliziert, wie wir gleich sehen werden.

Das P-Register ist nicht so einfach zu verwenden, da es nicht Zahlenwerte, sondern Flaggen (Bitmuster) enthält. Im einzelnen bedeuten im P-Register:

BIT Nr.	WERT	FLAGGE	ABKÜRZUNG
0	1	Übertrag	C(arry)
1	2	Null	Z(ero)
2	4	Unterbrechung	I(nterrupt)
3	8	Dezimal	D
4	16	Abbruch	B(reak)
5	32	nicht benutzt	
6	64	Überlauf	V
7	128	Vorzeichen	N(egativ)

Um eine der Flaggen des P-Registers zu löschen, empfiehlt es sich, das ganze Register mit POKE 783,0 zu löschen. Umgekehrt muß man beim Setzen der Bits sehr aufpassen wegen der Unterbrechungsflagge I. Eine 1 in I entspricht dem Maschinen-Befehl SEI, der alle Interrupts ausschaltet, auch die der Tastatur-Abfrage, was natürlich sehr störend sein kann! Um alle Flaggen außer der Unterbrechungsflagge I zu setzten, muß POKE 783,247 eingegeben werden.

So, jetzt aber wird es Zeit für ein Beispiel, wie vor dem SYS-Befehl Parameter eingegeben werden können. In der Literatur wird immer das Beispiel gewählt, den Cursor auf eine bestimmte Position zu setzen, beziehungsweise seine Position abzufragen. Dazu gibt es eine Routine, die bei beiden Rechnern ab Speicherzelle 65520 beginnt.

Sie nimmt die Zahl, die im X-Register steht, und verwendet sie als Zeilennummer; die Zahl des Y-Registers nimmt

sie als Spaltennummer, setzt dann den Cursor an diese Stelle und bringt die beiden Werte in die Speicherzellen 209/210 und 211.

Unser Beispiel hat die Aufgabe, den Cursor in die 4. Spalte der 7. Zeile zu setzen, dort das Dollar-Zeichen hinzuschreiben und es rot zu färben.

```
5 PRINT CHR$(147)
10 POKE 783,0
20 POKE 781,6
30 POKE 782,3
40 SYS 65520
```

Nach Löschen des Bildschirms werden zuerst alle Flaggen des P-Registers gelöscht (Zeile 5). Dann kommt die Zeilennummer in das X-Register (Zeile 10) und die Spaltennummer in das Y-Register (Zeile 30). Nach dem Eingeben dieser Parameter können wir mit SYS auf die Routine springen.

```
50 ZEILE=PEEK(209)+256*PEEK(210)
60 ADRESSE = ZEILE + PEEK(211)
70 POKE ADRESSE,36
```

In den Speicherzellen 209/210 können wir jetzt (zur Übung) die Zeilennummer ablesen. Die Adresse der Cursorposition im Bildschirmspeicher erhalten wir durch die Addition der Zeilennummer mit dem Inhalt der Speicherzelle 211. Dorthin POKEn wir den Bildschirmcode des Dollarzeichens, nämlich 36 (Zeile 70).

```
80 SYS 59940
90 FARBE=PEEK(243)+256*PEEK(244)
100 POKE FARBE+PEEK(211),2
```

Für das Färben des Dollarzeichens verwenden wir eine weitere Routine des Betriebssystems, die ab 59940 - beim VC-20 ab 60082 - beginnt. Sie ermittelt die Zeilenposition des Cursors im Farbspeicher und bringt diesen Wert in die Speicherzellen 243/244, wo wir ihn abfragen können (Zeile 90). Die Adresse der Cursorposition im Farbspeicher setzt sich aus diesem Wert plus der Spaltennummer zusammen, die wir wieder der Speicherzelle 211 entnehmen. Auf diesen Platz POKEn wir den Farbcode 2 für Rot (Zeile 100).

So leicht ist das, wenn man die Routinen und die Aufgaben der Speicherzellen kennt.

Basic-Befehl Nr. 46 SYS
- der SYS-Befehl startet ein Maschinenprogramm, das an der hinter dem SYS-Befehl stehenden Adresse beginnt.
- werden mit dem SYS-Befehl Routinen des Betriebssystems aufgerufen, die aber einige Parameter bei der Ausführung brauchen, können diese über die Speicherzellen 780 bis 783 vor dem SYS-Befehl eingegeben werden

35.3. Der USR-Befehl

Ohne Zweifel gehört auch dieser Befehl zu den Mauerblümchen von BASIC, obwohl sein Name - eine Abkürzung von User (Verwender) - eigentlich genügend Anreiz bieten müßte.

USR hat im Grund genommen dieselbe Funktion wie SYS. Er springt nämlich aus einem Basic-Programm direkt in ein Maschinen-Programm, arbeitet dieses so lange ab, bis er den Befehl RTS findet. RTS entspricht dem Basic-Befehl RETURN und springt in das Basic-Programm zurück.

Bei SYS steht die Sprungadresse gleich hinter dem Befehl.

Bei USR muß die Sprungadresse zuerst in die Speicherzellen 785/786 gePOKEt werden.

Beispiel: Sprung auf 56524
mit SYS: SYS 56524
mit USR: POKE 785,204 (204+256*220=56524)
POKE 786,220
X=USR(Y)

Kein Wunder, daß USR selten benutzt wird - ist er doch durch das POKEn der Sprungadresse in Low-/High-Byte Darstellung aufgebläht.



Illustration Rolf Boyke

Das ist aber nicht unnütz, weil USR mehr Fähigkeiten hat als SYS. Im Hinblick auf die im Abschnitt 35.2. »Der SYS-Befehl« aufgezeigten Möglichkeiten des SYS-Befehls sollte ich besser sagen:
USR hat andere Fähigkeiten als SYS.

USR ist eine Mischung von SYS und FN. Letzterer ist der BASIC-Befehl zur Definition selbsterfundener Funktionen (Lektion 33).

Bei USR allerdings wird die Funktion als Unterprogramm in Maschinensprache geschrieben, auf die dann wie gesagt der USR-Befehl zur Ausführung springt. Der Pfiff dabei ist aber, daß Zahlenwerte in das Maschinenprogramm mitgenommen beziehungsweise Resultate aus ihm herausgeholt werden können.

Wie läuft das ab: Das Argument Y, das in der Klammer hinter dem Befehl steht, wird zuerst in den Gleitkomma-Akkumulator Nr. 1 (FAC 1) in den Speicherzellen 97 bis 102 gebracht. Als Gleitkommazahl wird es vom angesprungenen Maschinenprogramm weiterverarbeitet. Das Resultat kommt dann wieder in den FAC 1 und steht als Wert von X zur Verfügung.

Das Argument Y kann übrigens auch ein komplexer Ausdruck sein, zum Beispiel:
X=USR(PEEK(A)+256*PEEK(B))

Ich möchte das an einem kleinen Beispiel demonstrieren.

Statt allerdings ein Maschinenprogramm selbst zu schreiben, verwende beziehungsweise springe ich auf eine Routine des Betriebssystems, die den Inhalt des FAC 1 für mathematische Operationen verwendet.

Als geeignete mathematische Operation habe ich die Routine für die Funktion INT gewählt, die im C64 ab der Adresse 48332 beginnt.

Zuerst definieren wir einen Wert für die Variable Y, der in die INT-Routine gebracht werden soll.

10 Y=14.35

Dann bestimmen wir die Sprungadresse für den USR-Befehl. Dazu teilen wir die Adresse 48332 auf in ein Low-Byte = 204 und ein High-Byte = 188. Diese POKEn wir nach 785/786

```
20 POKE 785,204
30 POKE 786,188
```

Jetzt folgt nur noch der USR-Befehl selbst und das Ausdrucken desResultats.

```
40 X=USR(Y)
50 PRINT X
```

Nach RUN erhalten wir das Resultat 14, wie das Gesetz für INT es befiehlt.

Sie können zur Übung statt INT auch COS verwenden, indem Sie auf die Adresse 57938 springen. Der Vergleich mit dem Befehl COS Y muß dasselbe Ergebnis bringen.

Wer hat übrigens gemerkt, daß wir überhaupt nichts mit der Speicherzelle 784 gemacht haben, obwohl sie doch angeblich am USR-Befehl beteiligt ist? Sie ist es wirklich, doch ohne unser Zutun. In diese Adresse wird beim Einschalten des Computers die Zahl 76 ($4C) geschrieben. Das ist der Code für den Maschinenbefehl »JMP« (JUMP), der dieselbe Wirkung hat wie GOSUB.

Bei Ausführung von USR springt nämlich die entsprechende Routine zuerst auf die Speicherzelle 784, findet dort den Sprungbefehl und in den beiden nachfolgenden Speicherzellen 785 und 786 die Sprungadresse - und führt so den geplanten Sprung aus.

Ich finde, USR ist es wert, in Ihre Überlegungen mit einbezogen zu werden, besonders wenn Sie innerhalb Ihrer BASIC-Programme extrem schnelle Unterprogramme in Maschinensprache eingebaut haben. Diese sind mit USR ganz elegant aufrufbar. Ich denke da zum Beispiel an eine Abfrage der Joysticks.

Basic-Befehl Nr. 47 USR(X)
- dieser Befehl ruft von einem Basic-Programm aus ein Maschinensprache-Unterprogramm auf
- Dabei kann ein numerischer Ausdruck als Argument direkt mitgegeben werden

35.4. Die Status-Variable ST

Neben den Befehlen (wie PRINT) und den Funktionen (wie COS) hat Basic auch noch drei fest definierte Variable, nämlich

TI, TI$ und ST

TI und TI$ haben wir schon in Lektion 23 »Die Uhr des Computers« behandelt.

Von den dreien ist ST wohl am seltensten anzutreffen, Grund genug, hier ein wenig darüber zu berichten.

Die Variable ST gibt den Status nach der letzten Einbeziehungsweise Ausgabeoperation an, beschränkt allerdings nur auf Operationen mit der Datasette und dem an einem gemeinsamen Ausgang angeschlossenen Diskettenlaufwerk und Drucker.

ST enthält dabei den Inhalt der Speicherzelle 144. Jedes Bit dieses Registers hat eine eigene Bedeutung.

Kassette:
Bit 2 (Wert 4): kurzer Block
Bit 3 (Wert 8): langer Block
Bit 4 (Wert 16): Lesefehler (nicht korrigierbar)
Bit 5 (Wert 32): Pruefsummenfehler
Bit 6 (Wert 64): File-Ende
Bit 7 (Wert 128): Band-Ende

Diskette/Drucker:
Bit 0 (Wert 1): Fehler beim Schreiben
Bit 1 (Wert 2): Fehler beim Lesen
Bit 6 (Wert 64): Daten-Ende
Bit 7 (Wert 128): "Device Not Present"-Fehler

Alle nicht aufgeführten Bits sind nicht benützt.

Wichtig ist noch zu erwähnen, daß nicht nur die in der Tabelle gezeigten Zahlen für ST auftreten, sondern auch Kombinationen davon. So ergibt zum Beispiel ein zu kurzer Block (4) und ein gleichzeitig aufgetretener Prüfsummenfehler (32) einen Wert von 36.

Kassettenoperationen

Zuerst testen wir mit einem Datei-Programm auf »FILE-ENDE«. Geben Sie bitte folgendes Programm ein:

```
10 OPEN 1,1,1,"DATEI"
20 PRINT#1,"QWERTY"
30 CLOSE 1
40 END
```

Zur Erinnerung: Nach dem OPEN-Befehl folgt zuerst die Nummer der Datei (ich nehme hier 1), dann die Gerätenummer (1 = Datasette) und schließlich die Sekundäradresse (1 = Schreiben).

Jetzt kommt der Lesevorgang:

```
50 OPEN 2,1,0,"DATEI"
60 FOR K=1 TO 10
70 GET#2,A$
80 PRINT A$;ST
90 NEXT K
100 CLOSE 2
```

In Zeile 50 eröffnen wir wieder eine Datei (diesmal Nummer 2) für die Datasette, jetzt aber zum Lesen (Sekundäradresse = 0). Die Schleife der Zeilen 60 und 90 schreiben uns 10mal ein Zeichen (A$) und den Wert von ST auf den Bildschirm.

Jetzt geht es los. Mit RUN starten wir den ersten Teil des Programms. Nach dem Schreibvorgang und der READY-Meldung (nach Zeile 40) müssen Sie das Band zurückspulen und mit GOTO 50 ab Zeile 50 weiterfahren. Jetzt wird die Datei gelesen.

Wir erhalten untereinander die sechs Buchstaben von Zeile 20, daneben für ST lauter 0 bis zum Ende, dann allerdings erscheint eine 64. Das ist der in der Tabelle angegebene Wert von ST für »File-Ende«.

Da die FOR-NEXT-Schleife zu lang ist, schießen wir mit dem Lesen über das File-Ende hinaus. Normalerweise kennen wir natürlich die Länge einer Datei nicht. Deshalb ist es besser, mit GOTO zurückzuspringen und das File-Ende abzufragen.

Löschen Sie bitte Zeile 60 und 90 und fügen Sie als Rücksprung und Prüfung ein:

```
85 IF ST=64 THEN 100
90 GOTO 70
```

Statt nach ST können wir natürlich genausogut nach PEEK(144) fragen.

Ein erneutes GOTO 50 bringt das erwünschte Resultat.

Um den vorhin schon erwähnten »kurzen Block« zu sehen, müssen wir einen entsprechenden Fehler künstlich erzeugen.

Löschen Sie bitte den ersten Teil des Programms bis einschließlich Zeile 40. Wir behalten also nur den Leseteil ab Zeile 50. Dann laden wir dieses Programm (Band vorher am besten wieder zurückspulen) mit SAVE »DATEI« nicht als Datei, sondern als ganz gewöhnliches Programm. Wenn es geladen ist, bitte das Band wieder zurückspulen.

Mit RUN starten wir jetzt das Lese-Programm, welches eine Datei sucht, aber nur ein Programm findet, allerdings mit dem richtigen Namen. Natürlich findet es einen Fehler und wir erhalten als Ausdruck:

```
36 oder manchmal auch 4
64 64
```

Die Zahl 36 entsteht aus 32+4, das bedeutet Prüfsum-

menfehler + Block zu kurz. Danach folgt wie vorher das File-Ende.

Die normale Blocklänge entspricht der Länge des Bandpuffers, in den die Datasette einspeichert. Er ist 191 Byte lang. In unserem Fall war offenbar der Block nicht ganz voll.

Der Prüfsummenfehler tritt dann auf, wenn eine der Überprüfungen von Kassettenoperationen einen Fehler gefunden hat. Der Blockfehler, auch der des zu langen Blocks, interessiert wohl weniger. Aber ein durch die Prüfungen gefundener Fehler könnte, frühzeitig noch vor dem Ausstieg des Programms entdeckt, abgefangen und ausgemerzt werden.

Die Formel dafür, ins obige Programm eingebaut ist:

```
85 IF ST<64 OR ST>8 THEN ..(z.B. LIST)
```

Statt LIST kann man natürlich auch etwas anderes nehmen.

Diskettenoperationen

Bei dem Diskettenlaufwerk bedeutet ST=64 »DATEN-ENDE«, das ist etwa dasselbe wie bei der Datasette. Um es zu überprüfen, nehmen wir dasselbe Programm wie vorher, nur müssen wir die DATEI-Zeilen der Diskette anpassen. Das sieht dann so aus:

```
10 OPEN 1,8,1,"DATEI,S,W"
20 PRINT#1,"QWERTY"
30 CLOSE 1
40 END
50 OPEN 2,8,0,"DATEI,S,R"
60 FOR K=1 TO 10
70 GET#2,A$
80 PRINT A$;ST
90 NEXT K
100 CLOSE 2
```

Das Ergebnis sieht hier so aus:

```
64
66
66
```

Die 64 ist natürlich wie erwartet der Wert für Daten-Ende. Die 66 ist 64+2, entstanden dadurch, daß wir über das Daten-Ende hinausgelesen haben. Die 2 bedeutet »Fehler beim Lesen« (in den englischen Beschreibungen heißt es »Read Time Out«). Ähnliches gilt für ST=1, das bedeutet »Fehler beim Schreiben« (englisch: Write Time Out).

Wie bei der Datasette kann das Überlesen natürlich mit der Abfrage

```
IF ST=64 THEN .... und GOTO...
```

gestoppt werden.

Interessant ist noch der Status beim Fehler »DEVICE NOT PRESENT«, den wir dadurch erzeugen, daß wir ein Programm oder die Directory von der Diskette laden wollen, ohne daß dieses Gerät angeschlossen oder eingeschaltet ist. Nach der Fehlermeldung geben wir direkt ein:

PRINT ST oder PRINT PEEK(144)

und wir erhalten die Zahl 128.

Wie man allerdings in einem BASIC-Programm durch Abfrage von ST=128 die Fehlermeldung »Device Not Present« und den dann folgenden Programmabbruch vermeiden kann, bedarf einer gesonderten Maßnahme:

Es gibt zwei Speicherzellen - 768/769 -, in denen in Low-/High-Byte-Darstellung eine Adresse steht, auf die das Betriebssystem springt, wenn die Meldung »DEVICE NOT PRESENT« ausgegeben werden soll. Diesen Zeiger kann man so »verbiegen«, daß die Meldung nicht ausgegeben wird und daß das Programm einfach weiterläuft.

Normalerweise steht in 768 die Zahl 139, in 769 die Zahl 227. Verbogen wird der Zeiger durch eine 61 (185 geht auch). Dadurch zeigt die Adresse auf eine Speicherzelle des Betriebssystems, in welcher der Assembler-Befehl "RTS", das bedeutet Rücksprung, steht. Jetzt können wir ungestört den ST-atus abfragen, wir müssen allerdings den negativen Wert von ST, also -128 nehmen.

```
10 POKE 768,61          Fehlermeldung abschalten
20 OPEN 1,8,15          Gerät ansprechen
40 CLOSE 1
50 POKE 768,139         Fehlermeldung einschalten
60 IF ST=-128 THEN 100  Sprung bei ausgeschalteten
                        Gerät
70 PRINT" FORTSETZUNG"  weiter im Programm
80 END                  Ende der DEMO
100 PRINT" GERAET EINSCHALTEN"
110 GET A$:IF A$=""THEN 110 Warteschleife
120 GOTO 10             neuer Versuch
```


Drucker-Operation

Für den Drucker sieht das Listing fast genauso aus. Die einzige Änderung ist in den Zeilen 20 und 30:

```
20 OPEN 1,4
30 PRINT#1
```

Unter bestimmten Umständen kann das Verbiegen des Zeigers in 768 entfallen, wie einige der Zuschriften ergeben haben.

Ich möchte nach eigenen längeren Versuchen aber dafür plädieren, die Fehlermeldung immer abzuschalten, um nie in Schwierigkeiten zu kommen.

Vorsicht ist die Mutter der Weisheit.

(Dr. H. Hauck/og)

ROCKUS







Comic Ingo Stein



Illustration: Rolf Boyke

Die 1000 Nöte der Datenspeicherung

Die magnetische Datenspeicherung, sei es auf Kassette oder Diskette, ist eine heikle Sache, bei der hin und wieder Fehler auftreten können. In diesem Bericht klären wir die möglichen Fehler und ihre Ursachen bei Floppy und Datasette.

Die Datasette und die Floppy 1541 sind für Besitzer eines C64 die bekanntesten Medien, um Programme und andere Daten für längere Zeit zu speichern. Sie sind fast schon eine Selbstverständlichkeit geworden. Doch während wir mit diesen Geräten arbeiten, ist uns meist nicht bewußt, welche komplizierten Vorgänge hinter dem Begriff »magnetische Datenspeicherung« stecken. Solange alles ordnungsgemäß abläuft, ahnen wir nichts oder nur wenig von der komplexen Organisation, die uns das Speichern von Programmen auf Kassette oder Diskette ermöglicht. Treten jedoch erste Probleme auf, stellt man fest, daß das eigene Wissen über den C64, die Floppy 1541 oder die Datasette nicht ausreicht.

Das erste Problem, mit dem der Anwender konfrontiert wird, ist die große Anzahl der Fehler, die beim Arbeiten mit dem Computer auftreten können. Doch auch die Floppy 1541 besitzt viele Fehlermeldungen, die verstanden werden sollten. Selbst die Datasette birgt einige Geheimnisse, die für das Laden und Speichern von Daten von Bedeutung sind.

Die Fehlermeldungen der Datasette und der Floppystation sind das Thema dieses Berichts, wobei wir zunächst mit dem einfachen Speichermedium Datasette beginnen wollen.

Soeben haben Sie sich eine Programmkassette gekauft, die ein neues Spiel enthält. Voller Erwartung schalten Sie Ihren C64 ein und laden das Programm. Nach merkwürdig kurzer Zeit stoppt die Datasette unvermittelt, während anders als sonst die ungewöhnliche Meldung »?LOAD ERROR« auf dem Monitor zu lesen ist. Trotz intensiver Bemühungen, wie mehrmaligem Tippen von RUN, läßt sich das Programm nicht starten. Schließlich versuchen Sie, das Programm erneut zu laden, doch erhalten Sie stets die gleiche unangenehme Nachricht auf dem Bildschirm. Das Spiel scheint verloren.

Die eben beschriebene Erfahrung werden viele Computeranwender machen, die oft mit der Datasette arbeiten. Der »?LOAD ERROR« ist dabei eine der vielen Fehlermeldungen, die der Computer zur Verfügung stellt, um Ihnen mitzuteilen, daß bestimmte Vorgänge nicht ordnungsgemäß abgelaufen sind. Er tritt beim Betrieb der Datasette auf und zeigt an, daß das zu ladende Programm nicht einwandfrei von der Kassette gelesen werden kann. Die Ursachen können dabei vielfältiger Art sein, werden jedoch immer mit der schon bekannten Meldung angezeigt.

Unsicheres Speichermedium – Datasette

Für die genauere Lokalisierung eines Fehlers kann eine bereits vom Computer reservierte Variable Auskunft geben. Sie hat den Namen »ST« und wird als Statusvariable bezeichnet, da sie den Status, das heißt, den augenblicklichen Zustand einer Datenübertragung, anzeigt. Jedes der 8 Bit dieses Wertes hat dabei eine bestimmte Aufgabe, welche Sie in Tabelle 1, zusammengefaßt, vorfinden. Die Bits 2 bis 5 befassen sich speziell mit der Datasette.

Um Fehler bei der Speicherung so gut als möglich zu vermeiden, trifft der Computer einige Vorkehrungen. Wird ein Programm auf Kassette geschrieben, legt der C64 direkt hinter dem gespeicherten Programm eine komplette Kopie der Daten an, so daß das Programm eigentlich doppelt gespeichert wird. Liest der Computer die Daten anschließend wieder ein, lädt er die erste Version in den Speicher. Danach vergleicht er sie mit der Kopie. Stimmen einige Byte nicht überein, wird zunächst ein Versuch zur Korrektur

unternommen. Gelingt dies nicht, setzt der Computer das Bit 4 der Statusvariable (PRINT ST ergibt den Wert 16) und stoppt den Ladevorgang mit einem »?LOAD ERROR«.

Eine weitere Sicherheit bietet eine Prüfsumme. Während des Speicherns bildet der C64 eine Prüfsumme über die Programmdaten und legt sie zusätzlich auf dem Magnetband ab. Laden Sie nun ein Programm, wird aus den gelesenen Bytes ebenfalls eine Prüfsumme errechnet, die mit der gespeicherten übereinstimmen muß. Ansonsten ist wiederum ein »?LOAD ERROR« die Folge. Ein Test der Statusvariable gibt genauere Auskunft. Sie hat in diesem Fall den Wert 32 (Bit 5 gesetzt): ein Prüfsummenfehler.

Zusätzlich kann es vorkommen, daß während des Ladevorgangs besondere Bytemarkierungen nicht ordnungsgemäß erkannt werden. Diese benötigt der Computer zur genauen Synchronisierung beim Lesen von Daten, denn er muß stets den Überblick über den Anfang und das Ende der einzelnen Bytes behalten. Fehlerhafte Markierungen stiften Verwirrung, weil die Daten nicht mehr richtig identifiziert werden können. In diesem Fall hat die Statusvariable ST bei einem »?LOAD ERROR« die Werte 4 oder 8 (Bit 2 oder 3 gesetzt).

Die richtige Einstellung ist alles

Meist hat das Auftreten eines Ladefehlers eine recht einfache Ursache. Im Laufe der Zeit bildet sich eine Schmutzschicht auf dem Tonkopf der Datasette, die das Lesen und Schreiben von Daten behindern kann. Mit einem alkoholgetränkten Wattestäbchen oder einer Reinigungskassette ist eine Reinigung schnell und einfach vorzunehmen. Hat man damit keinen Erfolg, kann das Problem an einem weiteren Phänomen liegen.

Möglicherweise war die Stellung des Tonkopfes Ihrer Datasette bezüglich des Kassettenbandes beim Schreiben des Programmes mit SAVE nicht die gleiche, wie beim Lesen mit LOAD. Man kann dies besonders bei Kassetten beobachten, die nicht mit der eigenen Datasette beschrieben wurden, da die Tonköpfe der Datasetten von Commodore in der Regel die unterschiedlichsten Einstellungen vorweisen können.

Oft hilft hier ein wiederholter Ladeversuch mit einer etwas anderen Tonkopfjustierung. Das Verstellen läßt sich übrigens leicht an einer der kleinen Schrauben neben dem Tonkopf vornehmen. Eine perfekte Justage ist mit einer kleinen elektronischen Schaltung zu bewerkstelligen, die wir im Sonderheft 15 des 64'er-Magazins vorgestellt haben.

Erscheint nach mehreren Einstellungen jedoch weiterhin die gefürchtete Fehlermeldung, handelt es sich aller Wahrscheinlichkeit nach um eine Kassette mit schadhaftem Bandmaterial. Beim Kauf von Leerkassetten sollten Sie deshalb auf gute Qualität achten (hochwertiges »Lownoise«-Eisenoxidband genügt). Fehlerhafte Softwarekassetten mit fertig gespeicherten Programmen sollten beim entsprechenden Händler reklamiert werden.

Doch auch Ihr bestes selbstgeschriebenes Programm ist möglicherweise noch nicht verloren, wenn ein Fehler aufgetreten ist. Zeigt die Statusvariable die Werte 16 (Fehler bei Vergleich) oder 32 (Prüfsummenfehler) können mit etwas Aufwand und Glück die Daten dennoch gelesen werden, da das Programm aufgrund der Sicherheitskopie zweimal hintereinander auf der Kassette steht. Die Markierungsfehler (PRINT ST ergibt 4 oder 8) sind dagegen recht schwer oder gar nicht zu umgehen. Eine Rettung des Programmes ist hier sehr unwahrscheinlich. Fand der Lesefehler kurz vor Programmende statt, kann bei einem Basic-Programm zumindest der bereits geladene Teil in Sicherheit gebracht werden. Am Ende eines solch verstümmelten Programmes befindet sich ein meist undefinierbares Durcheinander an Werten, was sich bei LIST durch wirre Zeichen in einer Basic-Zeile bemerkbar macht. Ein sogenannter OLD-Befehl (oder auch RENEW: Rückgängigmachen von NEW) kann das »Wirrwarr« wieder in Ordnung bringen. Da der C64 diese Anweisung jedoch bei seinem Basic 2.0 nicht kennt, müssen Sie in diesem Fall auf eine Basic-Erweiterung zurückgreifen.

Vorbeugen mit VERIFY

Damit die bisher geschilderten Probleme nicht auftreten, ist es ratsam, ein gerade gespeichertes Programm sogleich mittels VERIFY auf seine Fehlerfreiheit zu untersuchen. VERIFY vergleicht das im Speicher befindliche Programm mit dem gespeicherten und zeigt eventuelle Unterschiede durch eine Fehlermeldung an. Wurde das Programm korrekt auf Band geschrieben, lautet die Meldung nach dem Vergleich »OK«. Ein »?VERIFY ERROR« bedeutet hingegen, daß Fehler bei der Speicherung oder beim vergleichenden Lesen auftraten. Das Programm sollte nochmals mit SAVE gespeichert werden. Liegt die Vermutung nahe, daß das Kassettenband schadhaft ist, verwenden Sie am besten eine neue Kassette.

Diese Sicherheitsvorkehrungen sind zwar mit einem großen Zeitaufwand verbunden, vermeiden aber meist Probleme mit den eben beschriebenen Lesefehlern und den Ärger über ein verlorenes Programm. Das Rekonstruieren eines zerstörten Programmes ist zudem sehr viel umständlicher.

Wer jedoch ganz sicher gehen will, sollte zur Speicherung seiner Programme keine Datasette verwenden. Abgesehen von der nicht gerade überwältigenden Arbeitsgeschwindigkeit dieses Gerätes (die sich zwar mit Fast-Tape-Programmen erheblich beschleunigen läßt) ist das Schreiben von Daten auf Kassetten keine besonders sichere Angelegenheit.

Komfort und Sicherheit mit der Floppy 1541

Wesentlich sicherer und schneller ist dagegen eine Floppystation, die die Datenspeicherung auf Disketten erlaubt. Abgesehen vom Geschwindigkeitsvorteil bietet sie einen besseren Bedienungskomfort und eine relativ hohe Aufzeichnungsdichte.

Die Floppy 1541 zeigt einen sehr viel komplizierteren Aufbau als die vergleichsweise primitive Datasette. Neben dem Laufwerk und der Mechanik zur Bewegung des Schreib-/Lesekopfes sind viele elektronische Bauteile im Gehäuse des Laufwerks untergebracht. Sie übernehmen die Steuerung der mechanischen Teile und koordinieren den Ablauf des Schreibens und Lesens auf der Diskette. Betrachtet man diese Schaltungen genauer, stellt man fest, daß sie ähnlich kompliziert sind wie die eines Computers, wobei wir auf einen wesentlichen Aspekt der Floppy 1541 zu sprechen kommen.

Anders als bei anderen Floppylaufwerken, die lediglich Daten lesen und schreiben können, kann die Floppy 1541 als »intelligente« Diskettenstation bezeichnet werden, da sie selbstständig Operationen durchführen kann, ohne vom Computer gesteuert werden zu müssen. Wie ein vollständiger Computer besitzt die Floppy 1541 einen eigenen Mikroprozessor (CPU), ein Betriebssystem, das DOS genannt wird und etwas RAM als Arbeitsspeicher für diverse Aktionen. Lediglich Bildschirm und Tastatur fehlen. Sie werden aber nicht benötigt, da die Kommunikation aus-

schließlich über Ihren C64 erfolgt. Das DOS (»Disk Operation System« = Diskettenbetriebssystem) erlaubt dies sogar auf recht einfache Weise, wie wir noch sehen werden.

Wie Sie vielleicht wissen, besitzt der C64 eine Vielzahl (genau 30) verschiedener Fehlermeldungen, die Sie darauf aufmerksam machen, daß zum Beispiel ein Fehler während des Programmablaufs aufgetreten ist. Die »intelligente« Floppy 1541 kennt ebenfalls 34 Fehlermeldungen, die dem Benutzer Ungereimtheiten bei der Arbeit mit einer Diskette mitteilen. Die Floppystation besitzt jedoch keinen Bildschirm, auf dem sie diese anzeigen kann, und so muß sie einen anderen Weg gehen. Ist ein Fehler, etwa beim Lesen eines Programms, aufgetreten, beginnt die sonst konstant leuchtende rote Leuchtdiode am Laufwerk unregelmäßig zu flackern, während ungewöhnliche »Klick«-Geräusche aus dem Inneren des Gehäuses zu vernehmen sind. Oftmals beschwert sich die Floppy 1541 mit einem unangenehmen Rattern, bis der Diskettenmotor schließlich anhält und die rote Leuchtdiode nun in einem regelmäßigem Rhythmus blinkt. Nach all diesen akustischen und optischen Zeichen weiß der Computeranwender, daß bei der Arbeit mit dem Diskettenlaufwerk ein Fehler erkannt worden ist, doch kann man noch nicht feststellen, um welche Art von Fehler es sich handelt.

Kommunikation mit der Floppy 1541

Zunächst müssen wir mit der Floppystation in Verbindung treten. Dies geschieht über den sogenannten Kommandokanal der 1541. Er dient in erster Linie zum Übermitteln von Befehlen an die Floppystation, denn als intelligentes Gerät versteht sie natürlich auch Kommandos, die sie prompt und selbständig ausführt. Wir wollen uns aber mit der zweiten Aufgabe des Kommadokanals beschäftigen. Zusätzlich wird er nämlich als »Fehlerkanal« verwendet, der die Status- oder Fehlermeldungen der Floppy 1541 an den Computer übermitteln kann. Ist uns bei der Bedienung der Diskettenstation ein Fehler unterlaufen, was wir an der blinkenden roten LED erkannt haben, können wir über diesen Kanal das Laufwerk nach der Ursache des Mißgeschicks befragen, das heißt, die exakte Fehlermeldung erhalten. Doch wie können wir den Fehlerkanal ansprechen?

Die Floppy 1541 besitzt insgesamt 16 Kanäle, die alle gewissen Aufgaben dienen. Sie können entweder zum Schreiben oder Lesen von Daten verwendet werden. Sie sind von 0 bis 15 durchnumeriert, wobei die Kanäle 2 bis 14 dem Anwender zur Verfügung stehen. Die Kanäle 0, 1 und 15 sind jedoch von der Floppystation reserviert, wobei Kanal 15 unser besagter Kommandokanal ist. Er läßt nur das Senden von DOS-Kommandos oder das Empfangen der Fehlermeldungen zu. Wollen wir diesen Kanal ansprechen, müssen wir uns jedoch mit dem nötigen Handwerkszeug vertraut machen. Bevor wir die Fehlermeldung lesen können, müssen wir den Kommandokanal aktivieren, was oft auch als Öffnen bezeichnet wird. Dazu verwenden wir den Basic-Befehl OPEN, den Sie vielleicht schon vom Arbeiten mit Dateien kennen. Für das Öffnen des Kommandokanals ist folgendes einzugeben:

```
OPEN 1,8,15
```

Damit richten wir eine Datei mit der Nummer 1 ein und adressieren die Floppystation mit der Gerätenummer 8. Die letzte Zahl der obigen Anweisung gibt schließlich die Kanalnummer an, die der Datei zugeordnet werden soll. Sie lautet 15, und das ist - wie wir schon wissen - der Kommandokanal.

Nun können wir zum Beispiel einen Befehl an die Floppy 1541 senden, der nach einer Prüfung auf korrekte Syntax ausgeführt wird. Das soll uns aber hier nicht weiter interessieren. Wir wollen nur Daten empfangen, nämlich die Fehlermeldungen, die uns das Laufwerk bereitstellt. Das ermöglichen uns die Anweisungen INPUT# oder GET#. Sie funktionieren ähnlich wie die Anweisungen INPUT und GET. Mit INPUT# können mehrere Zeichen einer Datei gelesen werden, während GET# nur ein Zeichen empfängt. Wie INPUT und GET dürfen beide Befehle nur in einem Programm und nicht im Direktmodus verwendet werden, weshalb wir jeweils eine Zeilennummer voranstellen:

```
10 OPEN 1,8,15
20 INPUT#1,FN,FM$,TR,SE: PRINT FN;FM$;TR;SE
```

Nach Beendigung der Arbeit muß der Kommandokanal wieder geschlossen, das heißt inaktiviert werden, was durch den Befehl CLOSE erfolgt. Schließen wir also unsere Datei mit der Nummer 1 wieder:

```
30 CLOSE 1
```

Wenn Sie die Zeile 20 unseres kleinen Programms betrachten, werden Sie erkennen, daß die Fehlermeldung der 1541 aus vier unterschiedlichen Teilen besteht. Der erste Wert, der über den Kommandokanal empfangen wird, ist die Nummer des aufgetretenen Fehlers (FN). Der darauffolgende String enthält die Meldung im Klartext (FM$), während die abschließenden Werte TR und SE eine nähere Lokalisierung des Fehlers erlauben. Ihre genaue Bedeutung wird im Laufe dieses Berichtes noch erläutert.

Ebenso kann die Meldung Zeichen für Zeichen mit GET# gelesen werden. Zeile 20 würde dann so aussehen:

```
20 GET#1,A$: PRINT A$;: IF ST <>64 THEN 20
```

Seltsamerweise tritt hier die uns schon bekannte Statusvariable ST auf. Sie hilft uns, das Ende der Meldung festzustellen. Liegen keine weiteren Daten von einem Gerät am seriellen Bus mehr an, nimmt ST den Wert 64 (Bit 6 ist gesetzt) an. Die vollständige Fehlermeldung wurde gesendet.

Nachdem Sie unser kleines Fehlerprogramm mit RUN gestartet haben, sollten folgende Zeichen ausgegeben werden:

```
0 OK 0 0
```

oder

```
00, OK,00,00
```

Erstere Meldung erhalten Sie bei der Programmversion mit INPUT#, während zweitere bei der Zeile 20 mit GET# ausgegeben wird; die Aussage ist jedoch die gleiche. Diese beiden Texte sind eigentlich keine Fehleranzeigen. Sie teilen uns lediglich mit, daß ein Befehl oder eine Aktion der Floppystation einwandfrei ausgeführt worden ist. Selbst wenn wir kein Kommando gesendet haben, wird nach Ablauf unseres Programms immer diese Statusnachricht auf dem Bildschirm erscheinen. Man kann sie als das »READY« der Floppy 1541 verstehen. Die rote Leuchtdiode blinkt dabei nicht. Lediglich nach dem Einschalten der Diskettenstation können wir eine andere Statusmeldung empfangen. Sie lautet

```
73, CBM DOS V2.6 1541,00,00
```

Damit »begrüßt« uns die Floppy 1541 und sagt uns, mit welcher Version des DOS gearbeitet wird. Tritt sie in Verbindung mit einer blinkenden roten Leuchtdiode auf, hat sie eine andere Bedeutung, die wir noch kennenlernen werden. Doch können wir noch eine weitere Statusmeldung empfangen, deren Ankunft nicht durch ein Flackern der Arbeits-LED angekündigt wird.

Das »Wegkratzen« von Dateien

Bei der Arbeit mit Disketten kommt es oftmals vor, daß man ein gespeichertes Programm nicht mehr benötigt. Nun könnte man das Programm durchaus auf der Diskette

belassen, jedoch würde dadurch unnötiger Speicherplatz belegt, der für andere Programme oder Daten verloren ginge. Das Programm muß also von der Diskette gelöscht werden. Als eifriger Anwender der Floppy 1541 haben Sie deshalb sicherlich das Handbuch nach einem geeigneten Befehl durchsucht und sind auf eine entsprechende Anweisungsfolge gestoßen:

```
OPEN 1,8,15
PRINT#1,"S:(Name)"
CLOSE 1
```

Haben Sie bisher diese etwas komplizierte Folge einfach als Befehl hingenommen, ohne über die seltsamen Anweisungen nachzudenken, werden Sie spätestens bei der Erklärung des Kommandokanals in diesem Bericht eine Ähnlichkeit zu unserem Fehlerprogramm festgestellt haben. In beiden Fällen muß der Kommandokanal geöffnet werden. Beim Löschen eines Programmes von Diskette wird lediglich mit PRINT # eine Zeichenfolge an die Floppy 1541 geschickt und zwar ein DOS-Befehl mit dem Namen »S«. Er ist die Abkürzung für »SCRATCH«, was auf Deutsch etwa »wegkratzen« bedeutet. Nachdem man den Scratch-Befehl eingegeben hat, beginnt das Laufwerk zu arbeiten (rote Leuchtdiode leuchtet). Nach kurzer Zeit erlischt die rote LED wieder; die Floppy 1541 hat die Datei mit dem angegebenen Namen gelöscht. Als Bestätigung kann man eine Statusmeldung empfangen, die uns mitteilt, daß eine Datei gelöscht wurde:

```
01,FILES SCRATCHED,01,00
```

Die erste Zahl hinter der Klartextmeldung gibt an, wie viele Dateien gelöscht wurden. Dies ist durchaus sinnvoll, da man mit Hilfe der Jokerzeichen (? und *) mehrere Dateien zugleich von der Diskette entfernen lassen kann. Die Anweisung

```
OPEN 1,8,15
PRINT#1,"S:TE*"
CLOSE 1
```

richtet sich beispielsweise an sämtliche Dateien einer Diskette, die mit den Buchstaben »TE« beginnen. Befinden sich vielleicht die Programme »TEST« und »TEXT« auf der eingelegten Diskette, werden beide gelöscht, und bei der Statusmeldung »FILES SCRATCHED« die Zahl 02 (2 Dateien) ausgegeben.

Die »echten« Fehlermeldungen

Kommen wir aber nun zu den Meldungen, die einen richtigen Fehler anzeigen. Jede besitzt eine Nummer zwischen 0 und 74. Doch nicht jeder Nummer ist einer Fehlermeldung zugeordnet (es gibt »nur« 34). Wir werden diese im folgenden auch nicht nach Nummern sortiert vorstellen, sondern wir gruppieren sie nach ihrer Funktion und Bedeutung.

Beginnen wir gleich mit einigen einfachen Fehlern, die Sie bei der Arbeit mit der Floppy 1541 relativ häufig erhalten werden. Sie sind jedoch vergleichsweise harmlos.

Es kann vorkommen, daß Sie versehentlich mit dem Diskettenlaufwerk arbeiten wollen, obwohl sich keine Diskette darin befindet. Die Floppy 1541 bemerkt dies recht schnell und antwortet sofort:

```
74,DRIVE NOT READY,00,00
```

Sollten Sie aber dennoch eine Diskette in den Laufwerkschacht geschoben haben, kann die Ursache an einer unformatierten Diskette oder an Problemen bei der Schreib-/Lesekopfbewegung liegen. Im ersten Fall muß die Diskette formatiert werden (siehe Handbuch). Im zweiten Fall hilft manchmal die nachstehende Befehlsfolge:

```
OPEN 1,8,15
PRINT#1,"I"
CLOSE 1
```

Befindet sich unter Ihrer Diskettensammlung auch eine Diskette, die auf einer anderen Commodore-Floppystation (zum Beispiel CBM 3040) beschrieben wurde, kann die Floppy 1541 sie unter Umständen nur lesen. Beim Schreiben gibt es Schwierigkeiten; die Diskette wird sofort als fremd erkannt. Eine uns schon bekannte Meldung wird über den Kommandokanal gesendet:

```
73,CBM DOS V2.6 1541,00,00
```

Nun blinkt die rote LED und zeigt uns einen Fehler an. Die Nachricht soll uns daran erinnern, daß wir nur Disketten verwenden dürfen, die von einer Floppy 1541 beschrieben wurden oder zu diesem Diskettensystem passen.

Endlich liegt die richtige Diskette im Laufwerk. Voller Stolz wollen Sie ein selbstgeschriebenes Basic-Programm mit SAVE speichern. Doch schon nach kurzer Zeit blinkt erneut die rote Leuchtdiode. Auf dem Fehlerkanal wartet folgende Meldung auf ihren Abruf:

```
26,WRITE PROTECT ON,00,00
```

Hier ist offensichtlich die Schreibschutzkerbe der Diskette (die kleine eckige Aussparung an der Seite des Diskettenmantels) mit einer Plakette überdeckt. Wie der Name schon sagt, kann man durch Überkleben dieser Kerbe eine Diskette vor dem Überschreiben oder Löschen schützen. Entfernt man die Abdeckung wieder, ist die Speicherung wieder möglich.

»SYNTAX ERROR« gleich sechsmal

Haben Sie den Aufkleber von der Kerbe entfernt, können Sie mehrere Programme mit SAVE auf die Diskette schreiben. Irgendwann ist jedoch die Kapazität der Diskette erschöpft. Paßt ein Programm nicht mehr darauf, wird die Floppy 1541 die Nachricht


```
72,DISK FULL,00,00
```

auf dem Fehlerkanal bereitstellen. Dieser Fehler tritt auch auf, wenn Sie zu viele verschiedene Programme speichern wollen. Auf einer Diskette können nämlich maximal 144 Dateien abgelegt werden, selbst wenn ansonsten noch Speicherplatz zur Verfügung steht.

Betrachten Sie Tabelle 2, dann sehen Sie alle bisher besprochenen Status- und Fehlermeldungen. Eine Kurzbeschreibung gibt nochmals einen Überblick über deren Bedeutungen und Ursachen.

Kommen wir nun zu einem Fehlertyp, den Sie wahrscheinlich von Ihrem C64 kennen: dem »SYNTAX ERROR«. Er erscheint auf Ihrem Computer beispielsweise, wenn Sie einen Befehl eingetippt haben, den der C64 nicht kennt. Eine andere Ursache kann sein, daß eine Anweisung falsch, das heißt, nicht den Regeln entsprechend, eingegeben wurde. Der Computer beschwert sich mit einem »SYNTAX ERROR«, da die Syntax (Schriftform) eines Befehles nicht ordnungsgemäß befolgt wurde. Ähnlich verhält es sich auch bei der Floppy 1541. In vorangegangenen Abschnitten haben wir erwähnt, daß die Floppystation verschiedene Kommandos verarbeiten kann. Wie beim C64 muß bei der Eingabe, also dem Senden über den Kommandokanal eine Syntax eingehalten werden. Wird diese ignoriert, antwortet das Diskettenlaufwerk mit einem »SYNTAX ERROR«. Doch macht die Floppy 1541 bei der Art der Fehlerkennung genaue Unterschiede. Sie kennt insgesamt sechs Syntax-Fehlermeldungen mit den Fehlernummern 30 bis 34 und 39. Einige davon haben jedoch sehr ähnliche Bedeutungen.

Ein »SYNTAX ERROR« liegt vor, wenn ein Kommando, das über den Kommandokanal gegeben wird, nicht als Befehl identifiziert werden kann. Oftmals sind falsch angeordnete Parameter, die mit dem Befehl gesendet werden müssen, die Übeltäter. In diesem Fall wird die Floppystation

einen Syntax-Error mit der Nummer 30 ausgeben. Existiert der Befehl jedoch gar nicht, beschwert sich das Laufwerk mit einem Fehler der Nummer 31. Ist man aber fest davon überzeugt, einen korrekten Befehl eingegeben zu haben, kann der Grund bei einem Leerzeichen liegen, das man versehentlich vor das Kommandowort gesetzt hat. Das Diskettenlaufwerk akzeptiert nur Anweisungen, die an erster Position stehen. Die dritte Meldung, die auf einen falschen Befehl hinweist, ist der Syntax-Error mit der Nummer 39. Er tritt auf, wenn ein Befehl von der Floppy 1541 nicht eindeutig interpretiert werden kann, weil er zum Beispiel zusammen mit anderen Zeichen in einer Kommandozeile steht.

Eine sogenannte Befehlszeile, das heißt, die Befehlsfolge, die über den Kommandokanal geschickt wird, darf bei der Floppy 1541 maximal 58 Zeichen lang sein. Bei den meisten DOS-Anweisungen wird man dieses Maximum nicht überschreiten, da die Anordnung der eventuell mitgesendeten Parameter fest vorgeschrieben ist. Einigen Kommandos können jedoch variable Parameterlisten angehängt werden. In einem solchen Fall kann es leicht geschehen, daß die Befehlszeile länger als 58 Zeichen wird. Ein »32, SYNTAX ERROR,00,00« ist die Folge.

Kennt man die wichtigsten Diskettenbefehle, lernt man schnell, welche davon einen oder mehrere Dateinamen benötigen. Es ist dann höchst unwahrscheinlich, daß man zum Beispiel beim Löschen einer Datei vergißt, den Namen der betreffenden Datei anzugeben. Doch das Fehlen eines einzigen Zeichens kann es für das DOS der 1541 unmöglich machen, die Bezeichnung zu erkennen. Ein kleines Beispiel soll dies demonstrieren:

```
OPEN 1,8,15
PRINT#1,"STEST"
CLOSE 1
```

Wie Sie sehen können, fehlt hier der Doppelpunkt zwischen Befehl (S) und Dateiname (TEST). In diesem Fall kann die Floppy 1541 den Namen »TEST« nicht entziffern und gibt einen besonderen Fehler mit der Nummer 34 aus. Er weist uns darauf hin, daß ein Dateiname in der Kommandoanweisung fehlt, der für die Bearbeitung des angegebenen Befehls notwendig ist.

Der sechste »SYNTAX ERROR« steht ebenfalls mit Dateinamen im Zusammenhang. Seine Nummer ist 33. Tritt dieser Fehler auf, wissen wir, daß wir eines der Jokerzeichen falsch angewendet haben. Die Joker (? und *) sind bekanntlich Hilfsmittel, um Dateinamen abzukürzen, oder wie bei dem SCRATCH-Kommando, um mehrere Dateien einer Diskette gleichzeitig anzusprechen. Bei einigen Befehlen, die Namen als Parameter benötigen, dürfen die Jokerzeichen nicht eingesetzt werden. Eines dieser Kommandos ist Ihnen geläufig. Es ist das NEW- oder N-Kommando zum Formatieren einer Diskette. Mißbrauchen wir nun einmal vorsätzlich den Joker »*«:

```
OPEN 1,8,15
PRINT#1,"N:TESTDIS*,TD"
CLOSE 1
```

Die Diskettenstation wird dies nicht akzeptieren und folgende Fehlermeldung erzeugen:

```
33, SYNTAX ERROR,00,00
```

Womit wir den letzten der besprochenen Syntax-Fehler provoziert hätten. In Tabelle 3 haben wir alle Syntaxfehler mit kurzen Erläuterungen zusammengefaßt.

Stolpersteine bei der Dateiverwaltung

Neben Programmen kann die Floppy 1541 auch andere Daten auf einer Diskette speichern. Wenn Sie zum Beispiel ein Programm geschrieben haben, das die Adressen Ihrer Freunde verwaltet, sollten deren Daten für längere Zeit gespeichert werden, denn der Speicherinhalt des C64 wird beim Ausschalten gelöscht. Hierfür bietet die 1541 eine besondere Lösung: die sequentielle Datenspeicherung. Die Diskettenstation erlaubt es Ihnen, beliebige Daten in sequentiellen Dateien (SEQ-Dateien) abzulegen, um sie bei Bedarf jederzeit wieder in den Speicher des Computers zurückzuholen. Die Daten werden dabei der Reihe nach (sequentiell) auf die Diskette geschrieben und können ebenso wieder gelesen werden. Die Handhabung ist denkbar einfach. Dennoch können dem Programmierer einige Mißgeschicke unterlaufen.

Um die Fehlermeldungen, die die Floppy 1541 als Hilfestellung bereitstellt, zu erläutern, werden wir nun eine sequentielle Datei eröffnen. Selbst wenn Sie noch keine Erfahrung mit der Datenspeicherung auf der Floppy 1541 gemacht haben, werden Sie den Gedankengängen leicht folgen können, da wir die Erklärungen so allgemein wie möglich formulieren, ohne speziell auf Befehlsstrukturen einzugehen.

Die Handhabung einer Datei gliedert sich in vier Schritte. Sie erfolgt auf ähnliche Weise wie die Bedienung des Kommandokanals:

1. Öffnen der Datei mit OPEN
2. Schreiben von Daten mit PRINT#
3. oder Lesen von Daten mit INPUT# oder GET#
4. Schließen der Datei mit CLOSE

Nachdem wir uns einen Namen für unsere Datei ausgesucht haben (er soll »TESTDATEI« lauten), kann die Datenspeicherung beginnen. In unserer Zerstreutheit übergehen wir Punkt 1 der Liste und versuchen sofort, Daten auf die Diskette zu schreiben. Dieser äußerst gravierende Mangel fördert schon die erste Fehlermeldung zutage. Ohne daß der Fehlerkanal der Floppystation nach einer Fehlermeldung abgerufen werden muß, beschwert sich der Computer mit einem »?FILE NOT OPEN ERROR«, da wir die Datei nicht mit OPEN geöffnet haben. Der Fehlerkanal der Floppy 1541 zeigt das gleiche Resultat:

```
61, FILE NOT OPEN,00,00
```

Die Datei muß zunächst unter einem Namen eröffnet werden. Unglücklicherweise befindet sich auf unserer Diskette bereits eine andere Datei mit dem gleichen Namen. Die Organisation der Diskette erlaubt jedoch zwei Dateien gleichen Namens nicht, und so werden wir mit dem nächsten Fehler konfrontiert. Die Floppystation meldet:

```
63, FILE EXISTS,00,00
```

Ein neu gewählter Name (zum Beispiel »TESTFILE«) löst dieses Problem. Die gewünschten Daten können nach all den anfänglichen Schwierigkeiten endlich auf die Diskette geschrieben werden.

Hat man die besagten Daten in einer sequentiellen Datei abgelegt, ist es besonders wichtig, den Abschluß der Arbeit mit dem Befehl CLOSE anzukündigen. Die Datei muß wieder geschlossen werden. Ungeachtet dieser Tatsache schalten wir jedoch unseren Computer ab, ohne den letzten notwendigen Schritt zu tun. Die Datei wird damit nicht ordnungsgemäß geschlossen.

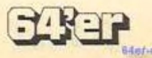
Schließlich wollen wir die Daten wieder in den Computer lesen. Versehentlich legen wir eine andere Diskette in das Laufwerk, die unsere Datei nicht enthält. Beim Öffnen der Datei kann die Floppystation die gesuchten Daten nicht finden und antwortet mit einer Fehlermeldung:

```
62, FILE NOT FOUND,00,00
```

Die Datei wurde nicht gefunden. Parallel dazu gibt der C64 ebenfalls die gleiche Meldung aus:

```
?FILE NOT FOUND ERROR
```

Auch beim Laden von Programmen kann dieser Fehler auftreten, denn Programme sind eine besondere Art von Dateien (PRG-Dateien). Haben wir nach längerem Suchen endlich die richtige Diskette in das Laufwerk eingescho-



ben, können wir einen erneuten Versuch wagen. Leider geben wir dabei den falschen Filetyp an, denn die Floppy 1541 kennt neben den Programmen (PRG) und den sequentiellen (SEQ) Dateien noch weitere Arten der Datenspeicherung. So kann man auch eine »User«-Datei (USR) eröffnen, die sich in der Handhabung jedoch nicht von SEQ-Dateien unterscheidet, und es gibt noch die relative Datenspeicherung (REL). Wir werden gleich auf sie eingehen.

Jedes Mal, wenn wir eine Datei öffnen, um Daten zu schreiben oder zu lesen, müssen wir den entsprechenden Dateityp angeben. Sie erinnern sich, daß unsere Datei vom Typ SEQ war. Wollen wir wieder darauf zugreifen, ist dieser Typ wiederholt anzugeben. Geschieht dies nicht, reagiert das Floppylaufwerk mit einem Fehler. Er kann provoziert werden, wenn wir unsere Datei mit dem Namen »TESTFILE« nicht als SEQ- sondern beispielsweise als USR-Datei ansprechen wollen. Die Antwort des Laufwerks folgt nach kurzer Zeit:

```
64, FILE TYPE MISMATCH,00,00
```

Doch nun wollen wir unsere Datei ordnungsgemäß mit richtigem Filetyp und Namen zum Lesen von Daten öffnen. Obwohl wir alle Bedingungen beachtet haben, beginnt die rote Leuchtdiode erneut zu blinken. Eine weitere Meldung wartet auf dem Fehlerkanal:

```
60, WRITE FILE OPEN,00,00
```

Mit Schrecken erinnern wir uns, daß wir beim Anlegen der Datei das Schließen mit CLOSE vergessen haben. Sie wurde damit nicht ordnungsgemäß abgeschlossen und kann jetzt nicht mehr gelesen werden. Die obige Nachricht weist uns unmißverständlich darauf hin. Die ganze Mühe, die wir uns mit unserer Datei machten, war also umsonst. Mit viel Aufwand und guten Programmierkenntnissen kann man zwar eine Datei nachträglich schließen und auf diese Weise die verloren geglaubten Daten retten, doch ist dies ziemlich zeitaufwendig.

Es ist sehr unwahrscheinlich, daß man all die Torturen, die wir eben spielerisch demonstriert haben, bei der Arbeit mit einer Datei in dieser Weise durchleben muß. Denn hat man sich etwas mit der Dateibedienung der Floppy 1541 beschäftigt, kann man viele dieser Fehler von Beginn an vermeiden.

Wir sind aber bei der Erläuterung der Fehlermeldungen zur Bearbeitung von Dateien noch nicht am Ende. In einem früheren Abschnitt dieses Artikels haben Sie kennengelernt, daß es neben der sequentiellen auch die relative Datenspeicherung gibt. Sie wird verwendet, wenn man schnell auf bestimmte Elemente der Daten zugreifen will. Diese Art der Datenspeicherung ist aber wesentlich komplizierter als etwa die sequentielle. Jedes Datenelement hat bei relativen Dateien eine begrenzte Länge und wird als Datensatz oder »Record« bezeichnet. Eine Datei kann nun viele dieser Datensätze enthalten, die alle aufsteigend numeriert werden. Hat man nun alle gewünschten Datensätze auf der Diskette gespeichert, kann man durch Angabe der Nummer den entsprechenden Datensatz direkt ansprechen, ohne etwa davor befindliche Daten überlesen zu müssen.

Hierbei können besondere Fehler auftreten, die die Floppy 1541 auch mit besonderen Meldungen bedacht hat. Die Nummern dazu lauten 50, 51 und 52.

Es kann beispielsweise vorkommen, daß man auf einen nicht vorhandenen Datensatz zugreifen will. Die Floppystation übermittelt uns dann die folgende Nachricht:

```
50, RECORD NOT PRESENT,00,00
```

Wir wissen somit, daß sich der Datensatz der angegebenen Nummer nicht in der Datei befindet. Schreibt man nun aber dennoch Daten hinein, ist die Floppy 1541 sehr hilfsbereit und erweitert die Datei um diesen zusätzlichen Datensatz. Beim Speichern von Datensätzen muß man jedoch sehr vorsichtig sein. Es wurde bereits erwähnt, daß alle Datensätze nur eine bestimmte Länge (Anzahl von Zeichen) haben dürfen. Ignoriert man diese Tatsache, sieht man sich mit einem weiteren Fehler konfrontiert:

```
51, OVERFLOW IN RECORD,00,00
```

Eine relative Datei kann jederzeit um beliebige Records erweitert werden. Irgendwann reicht aber die Speicherkapazität der verwendeten Diskette nicht mehr aus, um neue Datensätze aufzunehmen. Wenn es soweit ist, teilt uns die Floppystation dies durch eine Fehlermeldung mit:

```
52, FILE TOO LARGE,00,00
```

Die Datei kann nicht mehr erweitert werden, da sie die, für relative Dateien vorgeschriebene, Maximallänge erreicht hat. Diese Meldung gilt aber nur bei relativen Dateien und ist nicht zu verwechseln mit der »72, DISK FULL,00,00«-Nachricht, deren Bedeutung Sie schon kennengelernt haben.

Damit ist das Reservoir an Fehlermeldungen bezüglich der Dateibehandlung erschöpft. Sollte Ihnen bei der Fülle an Informationen schwindlig geworden sein, blicken Sie bitte auf Tabelle 4. Hier haben wir alle die in diesem Abschnitt besprochenen Fehlermeldungen noch einmal zusammengestellt.

Fehler des Schicksals

Alle Fehlermeldungen, die wir bisher bei unserer Exkursion durch die Welt der »Errors« erforscht haben, werden Ihnen im Laufe der Zeit oftmals begegnen, wenn Sie nur intensiv genug mit der Floppy 1541 arbeiten. Des weiteren zeichnen sie sich durch eine Gemeinsamkeit aus: Sie alle sind Anwenderfehler, das heißt Fehler, die auf das Unwissen oder die Unzulänglichkeit des Programmierers zurückzuführen sind. Hat man sich genügend Wissen über die Floppystation angeeignet, wird man sie größtenteils vermeiden können.

Auf unserem Pfad, der uns jetzt weg von Programmen und Dateien in das Innere des Diskettenlaufwerks führt, werden wir jedoch Fehlermeldungen kennenlernen, deren Ursache meist nicht durch eine Fehlbedienung gegeben ist. Hier sind oft minderwertige Disketten mit beschädigter Magnetoberfläche oder fremde Umwelteinflüsse die Übeltäter, die das ordnungsgemäße Speichern von Daten behindern und so für Fehler sorgen.

Hier ein Beispiel aus dem täglichen Leben: Sie laden eines Ihrer beliebtesten Spielprogramme und nehmen schon den Joystick zur Hand. Doch plötzlich gibt die Floppystation seltsame Geräusche von sich, während die rote Leuchtdiode, die zuvor konstant leuchtete, unregelmäßig zu flackern beginnt. Schließlich ertönt ein scheußliches Rattern und das Laufwerk hält mit einer Fehlermeldung an, obwohl man alle Eingaben in den Computer korrekt vorgenommen hat. Das Spielprogramm kann nicht mehr geladen werden und scheint für immer verloren. Wir werden diese Art von Fehlern im folgenden besprechen.

Warum Formatieren?

Um ihre Bedeutung genau zu verstehen, müssen wir uns zunächst mit der Organisation einer Diskette bei der Floppy 1541 beschäftigen. Wir werden kennenlernen, auf welche Weise unser Diskettenlaufwerk Programme und andere Daten auf einer Diskette ablegt.

Wenn mehrere Programme auf einer Kassette gespeichert werden sollen, läuft dies nach einem einfachen Schema ab. Die Daten werden der Reihe nach auf das Mag-

netband geschrieben. Ist ein Programm zu Ende, kann gleich dahinter ein neues Programm gespeichert werden. Man kann dies fortführen, bis die Kassette abgelaufen ist. Durch Abspielen des gesamten Bandes oder manuelles Vor- und Zurückspulen läßt sich nach einem bestimmten Programm suchen, was allerdings sehr lange dauern kann.

Die »Innereien einer Diskette

Eine Diskette kann ebenfalls mehrere Programme enthalten. Das einfache Prinzip der Kassettenspeicherung ist aber auf der runden Magnetscheibe einer Diskette kaum zu verwirklichen. Des weiteren zeichnet sich ein Floppylaufwerk durch schnelle Zugriffszeiten auf alle Daten aus; das heißt, ein Programm, das irgendwo auf der Diskette abgelegt ist, kann durch Angabe seines Namens sofort gefunden und in den Speicher des Computers geladen werden. Dies bedarf einer geregelten Organisation, die es der Floppystation erlaubt, sich schnell und problemlos auf der Diskette zurechtzufinden. Zu diesem Zweck müssen sich auf der Magnetoberfläche Markierungen befinden, die dem Diskettenlaufwerk bei der Orientierung behilflich sind. Sie sind von Gerät zu Gerät verschieden und bestimmen das Format einer Diskette. Um die für die Floppy 1541 notwendigen Markierungen auf eine Diskette zu bringen, muß diese formatiert werden. Vielleicht wissen Sie, daß die Floppystation einige Zeit dazu benötigt (etwa 90 Sekunden). Sehen wir uns einmal kurz an, was dabei mit der Diskette geschieht.

Beim Formatieren wird die Magnetscheibe der Diskette in 35 (magnetische) Spuren aufgeteilt, die konzentrisch um das Mittelloch angeordnet sind. Sie erhalten von außen nach innen die Nummern 1 bis 35. Während sich die Diskette dreht, kann der Schreib-/Lesekopf, von einem kleinen Motor geführt, auf jede dieser Spuren gelangen, um dort Daten zu schreiben oder zu lesen. Da eine solche Spur (englisch: Track) als Speichereinheit noch zu groß ist, um bequem bearbeitet werden zu können, wird sie in noch kleinere Untereinheiten zerlegt: die Sektoren. Ihre Anzahl kann von Spur zu Spur unterschiedlich sein, da deren Länge auf der Diskette von außen nach innen abnimmt. Pro Spur werden die Sektoren ab Null aufsteigend numeriert. Jeder Sektor kann somit durch eine Spur- und Sektornummer eindeutig bestimmt werden.

Auch innerhalb eines Sektors herrscht eine strenge Ordnung. Nach einem Vorspann (Kopf oder auch »Header«), der wichtige Daten über den entsprechenden Sektor enthält, folgt der Datenblock. Hier werden nun endlich die Daten abgelegt. Einen ausführlichen Kurs darüber finden Sie übrigens in unserem Floppy-Sonderheft 9/87. Jeder Sektor kann 254 Byte aufnehmen. Da ein Programm in der Regel länger ist, wird es beim Speichern auf mehrere Sektoren verteilt.

Anschließend wird beim Formatieren auf der Diskette ein Inhaltsverzeichnis angelegt, das die Namen der zukünftig gespeicherten Dateien aufnehmen wird. Daneben wird es wichtige Daten enthalten, die der Floppystation bei der Suche nach einer bestimmten Datei behilflich sind. Das Directory, wie das Inhaltsverzeichnis einer Diskette auch genannt wird, benötigt die gesamte Spur 18.

Man wird verstehen, daß bei all der komplizierten Organisation bereits ein geringer Fehler (oft genügt ein falsches Bit) größte Probleme ergeben kann. Wie bei der Speicherung auf der Datasette trifft die Floppy 1541 deshalb einige Vorkehrungen, um eventuelle Fehler möglichst zu vermeiden.

Wird zum Beispiel infolge eines SAVE-Befehles vom Computer ein Sektor auf einer Diskette mit Daten beschrieben, wird sogleich automatisch überprüft, ob alle Bytes ordnungsgemäß gespeichert wurden (VERIFY). Daneben werden, ähnlich wie bei der Kassettenspeicherung, Prüfsummen über die entsprechenden Daten erstellt und bei einem Lesezugriff mit den geladenen Bytes verglichen. Man findet jeweils eine Prüfsumme im Kopf und im Datenblock eines Sektors.

Man sieht, daß für die Datensicherheit gut gesorgt ist. Insbesondere das automatische, floppyinterne VERIFY beim Schreiben eines jeden Sektors verhindert, daß die Daten bereits fehlerhaft geschrieben werden. Doch trotz all dieser Vorkehrungen können Probleme bei der Datenspeicherung auf einer Diskette entstehen.

Obgleich die Ursachen eines Fehlers meist die gleichen sind (zum Beispiel fehlerhafte Disketten oder fremde Umwelteinflüsse), können sie in den verschiedensten Bereichen auftreten, die die Floppy 1541 sorgsam zu unterscheiden weiß. Man faßt sie allgemein unter dem Begriff »Lesefehler« zusammen, da durch irgendwelche Leseprobleme einer Diskette gewisse Daten oder Markierungen nicht korrekt identifiziert werden können. Die Floppy 1541 kennt insgesamt 9 dieser Lesefehler, von denen 6 die allgemeine Bezeichnung »READ ERROR« haben. Sie treten nur beim Lesen von Daten auf. Zwei Fehlermeldungen tragen den Namen »WRITE ERROR«, da sie nur beim Schreiben von Daten, wie zum Beispiel bei SAVE, in Erscheinung treten. Wir haben sie in sortierter Reihenfolge in Tabelle 5 abgedruckt. Dort finden Sie auch den neunten Lesefehler mit dem Namen »DISK ID MISMATCH ERROR«.

Diese Gruppe von Fehlern läßt sich in zwei Typen unterteilen: die »Soft« und die »Hard«-Errors, was Sie ebenfalls der Tabelle entnehmen können.

Obwohl alle Lesefehler größte Probleme bereiten, sind die Soft-Errors (»weiche« Fehler) in unserer Liste etwas harmloser als die Hard-Errors (»harte« Fehler). In den meisten Fällen können bei Fehlern dieses Typs die Daten des betroffenen Sektors noch relativ einfach und ohne gravierende Verluste gerettet werden. Ein »READ ERROR« der Nummer 22 weist beispielsweise darauf hin, daß der Header eines Sektors zwar einwandfrei gelesen werden konnte, die Floppystation aber den normalerweise folgenden Datenblock nicht gefunden hat. Die Meldung kann zum Beispiel lauten:

```
22, READ ERROR,14,09
```

Meist hat hier eine wichtige Markierung im Datenblock nicht den richtigen Wert, der Datenblock selbst ist aber korrekt. Mit etwas Programmieraufwand kann der Sektor noch gerettet werden, was uns an dieser Stelle aber nicht genauer interessieren soll.

Der »Fehlerteufel« schlägt zu

Bei den Fehlermeldungen bekommen nun auch die beiden Zahlen hinter dem Klartext, die bisher meistens auf Null standen, eine Bedeutung. Sie geben die Spur- und Sektornummer des Blocks an, in dem der Fehler aufgetreten ist. In unserem Beispiel ist es der Sektor 9 auf Spur 14. Wir werden diesen Sektor ab jetzt weiterhin als Beispiel verwenden.

Meldet die Floppystation

```
23, READ ERROR,14,09
```

handelt es sich ebenfalls um einen »Soft-Error«. Eine falsche Prüfsumme war hier die Ursache. Wie Sie schon wissen, enthält ein Sektor zwei Prüfsummen. Hier ist die Prüfsumme des Datenblocks gemeint, die mit der der gelesenen Datenbytes nicht übereinstimmt. In diesem Fall besteht die Hoffnung, daß lediglich die Prüfsumme falsch ist, die Daten jedoch korrekt geschrieben wurden. Einige Pro-

gramme, die sich mit der Diskettenbehandlung beschäftigen, wie zum Beispiel bestimmte Kopierprogramme, erlauben es deshalb, einen Sektor trotz eines »23, READ ERROR« zu lesen, um die Prüfsumme anschließend zu korrigieren. Meist sind jedoch fehlerhafte Datenbytes die Sünder. Die eben beschriebene Korrektur bleibt dann ohne Erfolg.

Ein Prüfsummenfehler im Sektor-Kopf kann ohne großen Aufwand nicht mehr repariert werden. Er hat einen harten Lesefehler zur Folge:

```
27, READ ERROR,14,09
```

Ein weiterer »Soft-Error« ist vom Typ »WRITE ERROR«. Er tritt bereits beim Schreiben von Daten auf. Wurde bei der Vergleichskontrolle eines eben gespeicherten Sektors ein Fehler in den Datenbytes entdeckt, meldet die Floppystation:

```
25, WRITE ERROR,14,09
```

Oftmals liegt dieser Meldung eine fehlerhafte Magnetbeschichtung der Diskette zugrunde. Doch ist der Fehler nicht besonders tragisch, da sich die Daten, zum Beispiel ein Programm, in der Regel noch im Speicher des Computers befinden. Man kann also einen weiteren Speicherversuch auf einer anderen Diskette vornehmen. Hat man auf diese Weise eine »Problemdiskette« erkannt, ist es ratsam, die restlichen Dateien von dieser Diskette auf einen neuen Datenträger zu kopieren. Vorsicht ist in solchen Fällen besser, als der Verlust sämtlicher Daten auf der schadhaften Diskette.

Harte Probleme mit »harten« Fehlern

Bei den »Soft-Errors« ist der Header (Kopf) eines Sektors in der Regel noch in Ordnung. Der Sektor kann auf der Diskette also noch gefunden werden. Handelt es sich jedoch um einen »Hard-Error« ist schon der Vorspann des betreffenden Sektors nicht mehr lesbar, wie es beispielsweise der Fehler mit der Nummer 20 anzeigt:

```
20, READ ERROR,14,09
```

Er besagt, daß die Floppy 1541 den Kopf eines Sektors nicht ausfindig machen konnte. Der Fehler kann sich dabei entweder auf nur einen Sektor oder eine ganze Spur beziehen. Ist die gesamte Spur betroffen, sind die darauf befindlichen Daten normalerweise für immer verloren, wie es auch beim »21, READ ERROR« der Fall ist. Hier ist es der Floppystation unmöglich, eine für das Lesen und Schreiben wichtige Markierung zu finden. Auch bei diesem Fehler braucht man sich keine Hoffnung auf eine Rettung der Daten machen.

Ein

```
29, DISK ID MISMATCH,14,09
```

tritt auf, wenn die »ID« (»IDentifizierung«) eines Sektors nicht mit der ID der Diskette übereinstimmt, die Sie beim Formatieren angegeben hatten. Bei diesem Fehler ist meist ein zerstörter Sektor-Header die Ursache.

Relativ selten ist der Lesefehler mit der Nummer 24 zu finden. Er bezieht sich auf die Codierung der Daten, bevor sie in einen Sektor auf der Diskette geschrieben werden. Aus organisatorischen Gründen werden die Daten nicht im Originalzustand gespeichert, sondern zuvor in einer gewissen Weise codiert. Nach dem Lesen müssen sie selbstverständlich wieder recodiert, das heißt zurückübersetzt werden. Im Normalfall entstehen hierbei keine Probleme. Im Fehlerfall können Werte auftreten, die nicht entziffert werden können. Die Antwort der Floppy 1541 lautet dann:

```
24, READ ERROR,14,09
```

Ebenso rar wie der Fehler 24 ist der zweite »WRITE ERROR« in der Tabelle 5. Er hat die Nummer 28. Hier wurde ein Sektor von einem davor befindlichen einfach überschrieben, so daß die Floppystation die Anfangsmarkierungen dieses Sektors nicht mehr finden kann. Dieser Fehler kann auftreten, wenn sich die Diskette zu schnell dreht, weil die Geschwindigkeitsregelung des Laufwerkmotors ausgefallen ist. Die Sektoren werden dann zu »lang« und überschreiben sich gegenseitig.

Bleiben zu guter Letzt noch einige Fehlermeldungen, die nicht so recht in eine unserer Gruppen passen wollen. Drei dieser Fehlermeldungen werden bei der normalen Anwendung der Floppy 1541 wohl nicht auftreten. Der fortgeschrittene Programmierer, der einige Direkteingriffe auf einer Diskette vornimmt, wird ihnen öfter begegnen. Das sind die Fehler mit den Nummern 65, 66 und 71, die wir hier nicht näher erläutern wollen. Sie sind aber in unserer Tabelle 6 dennoch aufgelistet. Lediglich der Fehler der Nummer 67 kann häufiger auftreten. Die Meldung lautet dann folgendermaßen:

```
67, ILLEGAL TRACK OR SECTOR,14,09
```

Da ein Programm meist länger ist als 254 Byte, muß es auf mehrere Sektoren verteilt werden. Damit die Floppystation nach dem Lesen eines Sektors weiß, in welchem Folgesektor das Programm oder die Datei fortgesetzt wird, enthalten die ersten zwei Byte des Datenblocks eines Sektors stets die Spur- und Sektornummer des darauffolgenden Programmteils. Durch unsachgemäßen Zugriff auf diesen Sektor ist es möglich, daß dieser Zeiger auf einen Sektor zeigt, der überhaupt nicht existiert. Spur 75 ist beispielsweise bei der Floppy 1541 nicht vorhanden, da die höchste Spur die Nummer 35 besitzt. Ein »ILLEGAL TRACK OR SECTOR«-Fehler entsteht. Wichtig ist dabei noch zu wissen, daß auch eine unerlaubte Sektornummer diese Fehlermeldung zur Folge hat.

An dieser Stelle ist es bestimmt kein »Fehler«, das Thema der Fehlermeldungen zu beenden. Manche Bereiche, die wir in diesem Bericht angeschnitten haben, werden sicherlich Fragen offenlassen. Zu ihrer Beantwortung weisen wir auf einschlägige Literatur wie »Die Floppy 1541« von Markt&Technik hin, die sich wesentlich intensiver mit diesem oder jenem Problem der Floppy 1541 auseinandersetzen kann. Die große Anzahl an Fehlermeldungen läßt die Komplexität der 1541 nur erahnen. Vielleicht ist das aber ein Ansporn, sich näher mit diesem interessanten Peripheriegerät zu beschäftigen. (Michael Thomas/ks)

Bit	Wert von ST wenn gesetzt	Bedeutung
0	1	Für die Datasette ohne Bedeutung
1	2	Für die Datasette ohne Bedeutung
2	4	zu kurzer Block
3	8	zu langer Block
4	16	Vergleichsfehler von ersten Daten und Kopie
5	32	Prüfsummenfehler
6	64	EOF; Ende der Datenübertragung
7	128	EOT; Bandende

Tabelle 1. Die Bitbelegung der Statusvariable ST

Nummer	Fehler	Bedeutung
00	OK	Befehl wurde ordnungsgemäß ausgeführt
01	FILES SCRATCHED,XX	XX Dateien wurden gelöscht
26	WRITE PROTECT ON	Diskette ist durch eine Schreibschutzplakette vor dem Überschreiben geschützt
72	DISK FULL	Diskette ist voll oder es wurden bereits 144 Dateien gespeichert
73	CBM DOS V2.6 1541	Einschaltmeldung; Diskette mit Fremdformat eingelegt

Tabelle 2. Ein Teil der Fehlermeldungen der 1541

Nummer	Fehler	Bedeutung
30	SYNTAX ERROR	allgemeiner Syntaxfehler
31	SYNTAX ERROR	ungültiger Befehl
32	SYNTAX ERROR	Befehlszeile zu lang
33	SYNTAX ERROR	unerlaubte Verwendung eines Jokerzeichens
34	SYNTAX ERROR	Dateiname ist nicht angegeben
39	SYNTAX ERROR	ungültiger Befehl

Tabelle 3. Die Floppy 1541 kennt sechs »SYNTAX ERROR«

Nummer	Fehler	Bedeutung
60	WRITE FILE OPEN	Zugriff auf eine nicht geschlossene Datei
61	FILE NOT OPEN	Datei ist nicht geöffnet
62	FILE NOT FOUND	Datei wurde nicht gefunden
63	FILE EXISTS	Datei mit angegebenem Namen existiert bereits
64	FILE TYPE MISMATCH	falscher Dateityp angegeben
Nur bei relativen Dateien		
50	RECORD NOT PRESENT	Datensatz der angegebenen Nummer existiert nicht
51	OVERFLOW IN RECORD	Datensatz zu lang
52	FILE TOO LARGE	Datei kann nicht erweitert werden, da Diskette voll

Tabelle 4. Die möglichen Fehler bei der Arbeit mit Dateien

Nummer	Fehler	Typ	Bedeutung
20	READ ERROR	hard	Blockheader nicht gefunden
21	READ ERROR	hard	Sync-Markierung nicht gefunden
22	READ ERROR	soft	Datenblock nicht gefunden
23	READ ERROR	soft	Prüfsummenfehler in Datenblock
24	READ ERROR	hard	Fehler bei der GCR-Recodierung
25	WRITE ERROR	soft	Fehler beim Verifizieren
27	READ ERROR	hard	Prüfsummenfehler im Sektor-Header
28	WRITE ERROR	hard	Block zu lang; nächster Block wurde überschrieben (Hardware-Defekt)
29	DISK ID MISMATCH	hard	falsche ID im Sektor-Header

Tabelle 5. Lesefehler – der Schrecken aller Programmierer. Nicht immer liegt es an einer defekten Diskette.

Nummer	Fehler	Bedeutung
65	NO BLOCK	Block bereits belegt
66	ILLEGAL TRACK OR SECTOR	Zugriff auf ungültige Spur oder Sektor
67	ILLEGAL TRACK OR SECTOR	Zeigerbytes in Datenblock zeigen auf ungültige Spur oder Sektor
70	NO CHANNEL	Kanal bereits belegt
71	DIR ERROR	zerstörte BAM

Tabelle 6. Außenseiter: Diese Lesefehler passen in keine der vier Gruppen

Floppyfehler abfangen

Manchem Programmierer sträuben sich die Haare, wenn er das Wort »Fehlermeldung« hört. Manchmal kann diese Einrichtung des Computers jedoch sehr sinnvoll angewendet werden.

Tritt während der Programmausführung ein Fehler auf, so bricht der Computer mit einer mehr oder weniger deutlichen Fehlermeldung ab. Für Fehler beim Zugriff auf die Diskette gilt dies jedoch nicht. Hier blinkt lediglich eine Leuchtdiode am Laufwerk und das Programm läuft oft weiter. Dies können wir für ein programmiertes Abfangen solcher Fehler ausnutzen. Allerdings benötigen wir dazu mehr Informationen. Das Floppylaufwerk hält diese für uns bereit – wir holen sie mit der Anweisung

```
10 OPEN 14,8,15:INPUT#14,A,B$,C,D:CLOSE 14
```

ab. Das Blinken am Laufwerk hört auf, und in den Variablen sind folgende Informationen gespeichert:

A: Nummer des Fehlers — C: Spur
B$: Fehlerbezeichnung im Klartext — D: Sektor

Jedem Fehler ist eine bestimmte Nummer zugeordnet. So hat die Meldung »File not found« die Nummer 62. »OK« und 0 sind Kennzeichen eines fehlerfreien Zugriffs auf die Diskette. Durch die Angabe der Spur- und Sektornummer erfahren wir außerdem, an welcher Stelle auf der Diskette der Fehler aufgetreten ist. Diese Angaben werden jedoch erst bei bestimmten Diskettenzugriffen interessant.

Beispiele für die Behandlung der häufiger auftretenden Fehler sind in unserem Programm (Listing 1) dargestellt.

Das Programm arbeitet mit sequentiellen Dateien: Eingabe und Speicherung von Adressen auf Diskette. Zu Beginn des Programms muß ein Dateiname angegeben werden. Befindet sich unter diesem Namen bereits eine Datei auf der Diskette, so werden alle gespeicherten Adressen auf dem Bildschirm ausgegeben. Das Programm ist nicht sehr komfortabel; dient aber auch nur zur Demonstration der Fehlerbehandlung.

Im Teil A (Zeile 1000 bis 1530) wird eine Tabelle mit allen Fehlernummern angelegt, auf die das Programm reagieren soll. Die Variable AF enthält die Anzahl der Nummern. Außerdem muß an dieser Stelle der OPEN-Befehl für das Lesen der Fehlermeldung stehen.

Teil E (ab Zeile 9000) holt die Fehlermeldung und vergleicht sie mit den gespeicherten in der Tabelle. Die ersten vier Fehler werden gleich hier abgefangen. Dies sind Fehler, deren Behandlung unabhängig vom Programm gleich sind. So wird auf das Fehlen einer Diskette im Laufwerk meist mit der Aufforderung, eine solche einzulegen, reagiert. Ist die Fehlernummer in der Tabelle nicht enthalten, bricht das Programm ab.

Dateien werden wie Programme mit ihrem Namen auf der Diskette gespeichert. Durch die OPEN-Anweisung in Teil B teilen wir dem Computer unter anderem mit, welche Datei wir lesen möchten. Interessant sind in diesem Programm nach der automatischen Fehlerbehandlung nur zwei Meldungen.

1. Fehlernummer 0:
Die Datei ist vorhanden. Sie wird daraufhin gelesen und die Adressen auf dem Bildschirm ausgegeben.
2. Fehlernummer 62:
Die Datei ist nicht vorhanden. Wir können gleich mit der Eingabe der Adressen über die Tastatur beginnen.

Alle anderen Fehler sind schon im Teil E abgefangen worden. Zur Überprüfung, ob diese Fehler beseitigt wurden, wird die OPEN-Anweisung erneut ausgeführt.

Teil C (ab Zeile 2200) befaßt sich mit der Eingabe der Adressen. Die Gestaltung dieses Teils ist abhängig von dem zu lösenden Problem. Wir haben deshalb dort keine besondere Fehlerbehandlung vorgesehen.

Mit der OPEN-Anweisung in Teil D (ab Zeile 2400) teilen wir dem Computer mit, unter welchem Namen die Adressen gespeichert werden sollen. Wiederum sind zwei Meldungen interessant.

1. Fehlernummer 0:

Die Datei ist nicht vorhanden. Wir geben die Adressen deshalb gleich aus und legen damit die Datei an.

2. Fehlernummer 63:

Die Datei existiert bereits. In diesem Fall geben wir ihr einen anderen Namen. Der alte Name wird lediglich um den Zusatz ».BAK« erweitert. Tritt dann bei der Speicherung der Adressen ein schwerwiegender Fehler auf (Stromausfall), so haben Sie immer noch die alte Datei (mit dem Zusatz).

Teil A und Teil E können Sie fast unverändert in Ihre eigenen Programme übernehmen. Ändern Sie den Inhalt von AF entsprechend der von Ihnen gewählten Anzahl von Fehlernummern. Die automatisch in Teil E abzufangenden Fehler müssen die ersten in der Liste der DATA-Zeilen sein. Programmieren Sie für jeden dieser Fehler im Teil E ein entsprechendes Unterprogramm und ergänzen Sie die Reihe der Sprungadressen in Zeile 9100. (pa)

```
1000 REM **************************** TEIL
     A ***                               <048>
1001 :                                   <215>
1010 AF=6          : REM ANZAHL FEHLERMELD
     UNGEN                               <089>
1020 DIM EN(AF)                          <203>
1100 DATA 26 : REM SCHREIBSCHUTZ         <213>
1110 DATA 72 : REM DISKETTE VOLL         <061>
1120 DATA 74 : REM KEINE DISKETTE IM LAUFW
     ERK                                 <175>
1130 DATA 64 : REM DATEITYP FALSCH       <115>
1140 DATA 62 : REM DATEI NICHT GEFUNDEN  <059>
1150 DATA 63 : REM DATEI EXISTIERT BEREITS <188>
1500 FOR I=1 TO AF                       <150>
1510 READ EN(I)                          <007>
1520 NEXT I                              <078>
1530 OPEN 14,8,15                        <146>
1540 :                                   <248>
1600 REM **************************** TEIL
     B ***                               <146>
1601 :                                   <053>
1610 DIM N$(300),S$(300),O$(100),T$(100) <092>
2000 AA=0             : REM ANZAHL ADRESSEN <034>
2100 PRINT CHR$(147) : REM BILDSCHIRM LOES
     CHEN                                <096>
2110 INPUT "DATEINAME: ";DN$             <028>
2120 PRINT                               <190>
2130 OPEN 1,8,8,DN$+",S,R"               <097>
2140 GOSUB 9000                          <128>
2150 IF A=62 THEN 2300 : REM DATEI NICHT D
     A                                   <143>
2160 IF A=0 THEN 2200  : REM DATEI VORHAND
     EN                                  <094>
2170 CLOSE 1                             <149>
2180 GOTO 2100          : REM NOCH MAL VERS
     UCHEN                               <052>
2190 :                                   <134>
2200 REM **************************** TEIL
     C ***                               <242>
2201 :                                   <145>
2210 INPUT#1,AA                          <135>
2220 FOR I=1 TO AA                       <103>
2230 INPUT#1,N$(I),S$(I),O$(I),T$(I)     <134>
2240 PRINT N$(I) : PRINT S$(I)           <165>
2250 PRINT O$(I) : PRINT T$(I)           <187>
2260 PRINT                               <074>
2270 NEXT I                              <066>
2300 CLOSE 1                             <023>
2310 AA=AA+1                             <080>
2320 INPUT "NAME: ";N$(AA)               <134>
2330 INPUT "STRASSE: ";S$(AA)            <140>
2340 INPUT "ORT: ";O$(AA)                <081>
2350 INPUT "TELEFON: ";T$(AA)            <159>
2360 PRINT                               <176>
2370 INPUT "WEITER MACHEN (J/N)";E$      <069>
2380 IF E$="J" THEN GOTO 2310            <136>
2390 :                                   <080>
2400 REM **************************** TEIL
     D ***                               <192>
2401 :                                   <091>
2410 OPEN 1,8,8,DN$+",S,W"               <133>
2420 GOSUB 9000                          <154>
2430 IF A=0 THEN 2600                    <081>
2440 CLOSE 1                             <165>
2450 IF A<>63 GOTO 2400                  <180>
2500 PRINT#14,"S:"+DN$+".BAK"            <090>
2510 PRINT#14,"R:"+DN$+".BAK="+DN$       <059>
```

```
2520 GOTO 2400                           <246>
2600 PRINT#1,AA                          <097>
2610 FOR I=1 TO AA                       <241>
2620 PRINT#1,N$(I):PRINT#1,S$(I)         <006>
2630 PRINT#1,O$(I):PRINT#1,T$(I)         <113>
2640 NEXT I                              <184>
2650 CLOSE 1 : CLOSE 14                  <150>
2660 PRINT "PROGRAMM BEENDET"            <155>
3000 END                                 <208>
3010 :                                   <192>
9000 REM **************************** TEIL
     E ***                               <192>
9001 :                                   <087>
9010 INPUT#14,A,B$,C,D                   <223>
9020 IF A=0 THEN RETURN                  <027>
9030 PRINT                               <242>
9040 I9=1                                <213>
9050 IF EN(I9)=A THEN 9100               <124>
9060 I9=I9+1                             <192>
9070 IF I9<=AF THEN 9050                 <212>
9080 PRINT A;" ";B$;"{2SPACE}";"SPUR:";C;"
      ";"SEKTOR:";D                      <248>
9090 STOP : REM PROGRAMMABBRUCH          <106>
9100 ON I9 GOTO 9200,9300,9400,9500      <081>
9110 RETURN                              <022>
9120 :                                   <206>
9200 REM *** FEHLERMELDUNG 1             <174>
9210 PRINT "DIE DISKETTE IST SCHREIBGESCHU
     ETZT" : PRINT                       <185>
9220 PRINT "WENN SIE MIT DER DISKETTE ARBE
     ITEN WOLLEN,"                       <043>
9230 PRINT "{2SPACE}DANN ENTFERNEN SIE DEN
     SCHUTZ"                             <072>
9240 GOSUB 9900                          <004>
9250 RETURN                              <164>
9260 :                                   <092>
9300 REM *** FEHLERMELDUNG 2             <028>
9310 PRINT "DIE DISKETTE ODER DAS INHALTSV
     ERZEICHNIS SIND VOLL"               <164>
9320 PRINT "{2SPACE}LEGEN SIE EINE ANDERE
     DISKETTE EIN"                       <169>
9330 GOSUB 9900                          <094>
9340 RETURN                              <254>
9350 :                                   <182>
9400 REM *** FEHLERMELDUNG 3             <136>
9410 PRINT "ES BEFINDET SICH KEINE FORMATT
     IERTE"                              <010>
9420 PRINT "{2SPACE}DISKETTE IM LAUFWERK" <071>
9430 GOSUB 9900                          <194>
9440 RETURN                              <098>
9450 :                                   <026>
9500 REM *** FEHLERMELDUNG 4             <246>
9510 PRINT "DIESE DATEI EXISTIERT BEREITS
     ALS PROGRAMMDATEI"                  <114>
9520 PRINT "{2SPACE}AUF DER DISKETTE"    <037>
9530 PRINT "WAEHLEN SIE EINEN ANDEREN NAME
     N"                                  <044>
9540 GOSUB 9900                          <050>
9550 RETURN                              <210>
9560 :                                   <138>
9900 REM *** TASTE ABWARTEN              <219>
9910 PRINT : PRINT "WENN FERTIG - TASTE DR
     UECKEN"                             <228>
9920 GET E$ : IF E$="" THEN 9920         <035>
9930 RETURN                              <080>
9940 :                                   <008>
```

Listing 1. Abfangen von Floppyfehlern beim Verwalten von Adressen

Leichter Einstieg in Geos

Liegt Ihrem neuen C64 eine Diskette mit der Aufschrift »Geos« bei und Sie wissen noch nicht recht, was Sie als Einsteiger damit anfangen sollen? Dann wird Ihnen dieser Artikel zeigen, wie Sie aus Geos den größtmöglichen Nutzen ziehen.

Leider läßt sich in einem einzigen Artikel nur ein erster Einblick in die Geos-Welt vermitteln. Um Geos bis ins letzte Bit auszureizen, ist schon erheblich mehr Platz erforderlich. Solche detaillierten Informationen finden Sie in unserem Geos-Kurs ab Ausgabe 2/87.

Durch diesen Artikel erhalten Sie eine genaue Vorstellung davon, inwiefern Ihnen Geos nützlich sein kann.

Die Bezeichnung »Geos«

Wie zahlreiche Produktbezeichnungen ist Geos eine Abkürzung. »Geos« steht für »Graphics Environment Operating System«, was man recht frei als »grafisches Betriebssystem und Benutzeroberfläche« übersetzen würde. »Grafische Benutzeroberfläche« bedeutet, daß man dem Computer keine Befehle im Klartext mitteilt (über Tastatureingaben), sondern anhand von grafischen Symbolen mit dem Joystick oder einem anderen Eingabegerät die gewünschten Funktionen auslöst. Der Vorteil ist unbestritten die leichtere Erlernbarkeit des Umgangs mit einem neuen Programm, denn von solchen Symbolen läßt sich zumeist auch ohne große Phantasie die Bedeutung ableiten. Ein Beispiel: Um Daten zu löschen, muß man im C64-Betrieb ohne Geos Befehlssequenzen

```
OPEN 15,8,15,"S:NAME":CLOSE 1
```

eintippen; unter Geos sehen Sie einen Papierkorb, in den man am Bildschirm die unerwünschten Daten »hineinwirft«.

Eine solche grafische Orientierung war bislang nur neueren Geräten wie dem Atari ST und der AMIGA-Serie vorbehalten. Dank Berkeley Softworks und Commodore steht nun mit Geos auch für den C64 eine solche Software zur Verfügung.

So startet man Geos

Da Geos nicht fest im C64 eingebaut ist, steht es unmittelbar nach dem Einschalten auch noch nicht im Speicher. Deshalb muß die Geos-Systemdiskette, die dem C64 beiliegt, in das Diskettenlaufwerk eingeschoben und folgender Befehl eingegeben werden:

```
LOAD ":*",8,1 <RETURN>
```

Auf diese Anweisung hin beginnt der Ladevorgang (bei Geos auch »Bootvorgang« genannt).

Bald erscheint der Text »BOOTING Geos« und – sofern alles geklappt hat – das Einschaltbild von Geos (Bild 1).

Sollte sich dieser Bildschirmaufbau jedoch nicht zeigen, sondern das System beim Laden »abstürzen«, so ist ein erneuter Bootversuch zu unternehmen. Diese kleine Unzulänglichkeit sollte jedoch nicht vom Geos-Gebrauch an sich abschrecken, da er alles andere als problematisch ist.

Zurück zum korrekten Laden. Dieses vollzieht sich dank des Geos-eigenen Turboladers innerhalb von zirka 30 Sekunden. Auch nach dem Laden von Geos bleiben die Turbo-Routinen aktiv, das heißt, Sie nutzen im Geos-Betrieb (beim Laden und Speichern auf Diskette) immer die deutlich erhöhte Floppy-Geschwindigkeit aus.

Die Bedienungselemente von Geos

Im folgenden wollen wir anhand dieses Einschaltbildes (Bild 1) die Geos-Bedienungselemente vorstellen. Lassen Sie dazu die Diskette im Laufwerk, da noch darauf zugegriffen wird.

Halten Sie sich bitte exakt an die gegebenen Anweisungen. So ist gewährleistet, daß Ihre Diskette keinen Schaden nimmmt.

Das elementarste Bedienungselement ist der blaue Pfeil (»Mauszeiger« oder »Mauscursor« genannt), den Sie ständig mit dem Joystick in Port 1 bewegen (probieren Sie es gleich aus, aber drücken Sie bitte nicht auf den Feuerknopf!). Auch andere Eingabegeräte als der Joystick (zum Beispiel Maus, Lightpen, Koala Pad) sind möglich (siehe Geos-Handbuch, Abschnitte 1.1–1.5).

Die Bedeutung des Mauszeigers liegt darin, daß Sie ihn immer auf das jeweilige Bedienungselement bewegen, welches Sie dann durch Betätigen des Feuerknopfes auslösen.

Die grafischen Bedienungselemente von Geos sollen Sie nun kennenlernen.

Icons (Kleingrafiken)

Viele Funktionen werden unter Geos durch eine Grafik repräsentiert. Am Einschaltbildschirm sind gleich mehrere Icons (so der Fachausdruck für Kleingrafiken) zu finden: Diskettensymbol (rechts oben), Drucker und Papierkorb (rechts unten) sowie acht Icons in der eingerahmten Mitte des Bildschirms, auf die wir später noch zu sprechen kommen.

Als erstes wollen wir das Diskettensymbol »anklicken«. Bewegen Sie dazu den Mauszeiger auf das Diskettensymbol und lösen Sie dann einmal kurz den Feuerknopf am Joystick aus. Das Diskettensymbol blinkt auf und es findet ein Zugriff auf die Diskette im Laufwerk statt, da dieses Icon eine Diskette initialisiert. Wenn Sie eine Diskette wechseln wollen, genügt es nämlich nicht, einfach die alte Diskette herauszunehmen und die neue einzulegen; zusätzlich ist noch das Disk-Symbol anzuklicken, wie wir es soeben getan haben. Legen Sie deshalb jetzt bitte eine andere Diskette (zum Beispiel C64-Demodiskette) in das Laufwerk und klicken Sie das Disketten-Icon an; an der Änderung des Bildschirmaufbaus und den hörbaren Floppyzugriffen erkennen Sie, daß Geos die neue Diskette akzeptiert hat.

Programme starten

Bislang haben wir uns im sogenannten »DeskTop« befunden. Dies ist das Programm, das nach dem Laden von Geos als erstes im Speicher steht und ausgeführt wird. DeskTop dient nur zur Verwaltung von Dateien und Programmen; diese werden von DeskTop aus aufgerufen. Der Aufruf vollzieht sich, indem am Bildschirm das Icon des gewünschten Programms angesteuert und »doppelgeklickt« wird (darunter versteht man das Drücken des Feu-

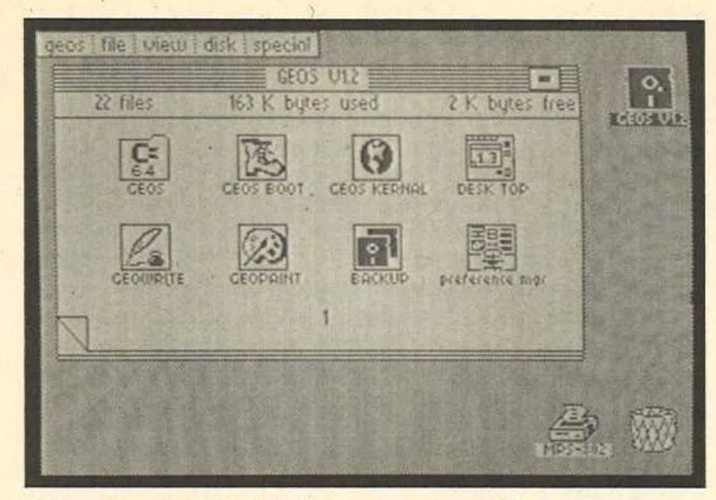


Bild 1. Einschaltbild nach dem Laden von Geos

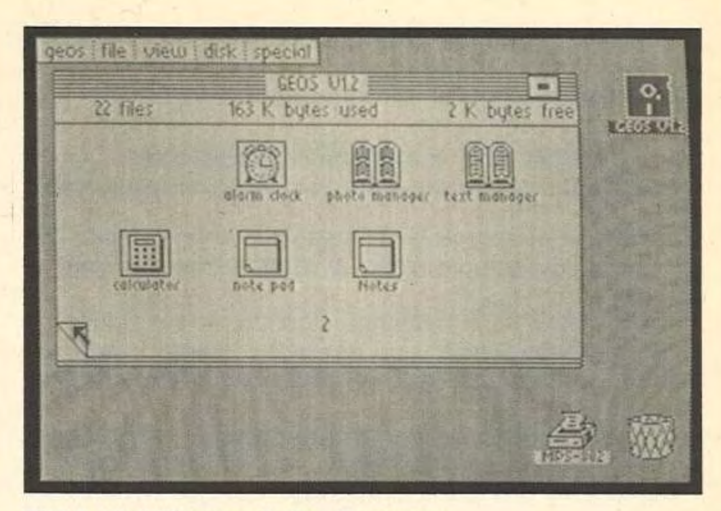


Bild 2. Umblättern mit dem Eselsohr

erknopfs zweimal kurz hintereinander). Als Beispiel wollen wir das Geos-Ladeprogramm von Geos aus starten. Sein Icon steht links oben im eingerahmten Kasten; es trägt die Aufschrift »C= 64«. Bewegen Sie nun den Mauszeiger darauf und drücken Sie zweimal hintereinander auf »Feuer«. Sie sehen, wie Geos den Ladebefehl zur Ausführung gibt und sich wieder selbst lädt. Natürlich hätten wir uns diesen erneuten Ladevorgang sparen können, aber als Beispiel eignet er sich recht gut.

Die Windowtechnik

Das gerade angeklickte Icon »C= 64« befand sich zusammen mit sieben weiteren Icons in einem eingerahmten Bildschirmbereich. Solche Bildschirmbereiche bezeichnet man als Fenster (englisch »window«). Der Vorteil von Fenstern liegt darin, daß sie nur einen Teilbereich des Bildschirms beanspruchen und sich überlagern können. Wenn ein Fenster nicht mehr benötigt wird, stellt Geos den Bildschirminhalt wieder her, der sich zuvor an der Window-Position befand. Dies werden wir bei der Erklärung der »Pull-down-Menüs« sehen.

Das Directory-Fenster soll uns zunächst als Beispiel für die grundsätzlichen Elemente von Windows dienen. Ein Fenster ist grundsätzlich durch einen Kasten eingerahmt. Eine Besonderheit des Directoryfensters stellt hingegen das links unten stehende »Eselsohr« zum Seitenumblättern dar. Wenn Sie die linke Hälfte des Eselsohr anklicken (einmal klicken!), wird eine Seite nach vorne geblättert, das heißt, Sie sehen die Icons der nächsten Programme auf der aktuellen Diskette (Bild 2).

Am wichtigsten erscheint jedoch das Schließsymbol rechts oben (das ausgefüllte Rechteck im Rahmen). Sobald dieses Symbol ausgelöst wird, schließt Geos das Fenster wieder. Dieses wird jetzt entweder neu aufgebaut oder es verschwindet vom Bildschirm, um Platz für neue Informationen zu machen.

Pull-down-Menüs

Eine ausgeprägte Menüsteuerung besitzen auch viele C64-Programme unabhängig von Geos. Insofern wird Ihnen das Prinzip von Menüs nicht unbekannt sein: Aus einer Vielzahl der zur Diskussion stehenden Funktionen, die durch einen kurzen Text (meist nur 1 Wort) beschrieben werden, wählt man am Bildschirm die gewünschte Option aus.

Links oben am DeskTop-Bildschirm findet man eine Aufstellung von Menüpunkten (»geos«, »file«, »view«, »disk«, »special«). Diese Menüpunkte wählt man durch Anklicken aus, also so, wie Sie es vom Disketten-Icon kennen.

Lösen Sie deshalb jetzt den Feuerknopf aus, während sich der Mauszeiger über dem Menüpunkt »geos« befindet. Es rollt sich dann ein weiteres Menü auf; daher kommt die Bezeichnung »Pull-down-Menü« (Herunterzieh-Menü).

Dieses Pull-down-Menü können Sie dadurch verlassen, indem Sie mit dem Mauszeiger aus seinem Bereich steuern; dann wird es unverzüglich vom Bildschirm entfernt.

Um hingegen eine Funktion anzuwählen, müssen Sie diese nur anklicken. Als Beispiel eignet sich »Geos info« oder »DeskTop info«, woraufhin jeweils ein Fenster mit Informationen über die Autoren von Geos und DeskTop erscheint. Durch Betätigen des Feuerknopfes verläßt man auch diese Fenster. Dies nur ein kleiner Nachtrag zur Windowtechnik.

Viele Bedienungselemente

Sie haben jetzt alle Geos-Bedienungselemente kennengelernt: Icons, Windows und Pull-down-Menüs. Zugegebenermaßen stellen diese eine Umstellung gegenüber dem normalen C64-Betrieb dar, aber daß sie eine hohe Bedienungsfreundlichkeit garantieren, haben Sie sicherlich bereits gemerkt.

Da diese Bedienungselemente in allen Geos-Programmen wiederkehren, ist eine Umstellung auf neue Programme einfach. Deshalb wollen wir uns an dieser Stelle nicht mit Details des DeskTop befassen, die Sie dem Handbuch oder dem Standardwerk »Alles über Geos« entnehmen können. Vielmehr wollen wir weitere Geos-Programme vorstellen, damit ein besserer Einblick in die Leistungsfähigkeit der Geos-Software entsteht.

Zuvor jedoch ein paar Anregungen, welche Menüpunkte zum Ausprobieren des DeskTop sehr empfehlenswert sind, weil man mit ihnen keinen Schaden anrichten kann:

Das »view«-Menü liefert eine sortierte Anzeige der Diskettenfiles; diese können jedoch nur aus »view by icons« heraus gestartet werden. »open« im »disk«-Menü entspricht dem Disketten-Icon, »basic« in »special« führt zum Zurücksetzen (Reset) des C64 und »reset«, ebenfalls in »special«, initialisiert Geos, verläßt es jedoch nicht.

Geos bietet Ihnen nicht nur die einzigartige Gelegenheit, den Umgang mit einer grafischen Benutzeroberfläche bereits auf dem C64 zu erproben und damit zukunftsträchtiges Basiswissen zu erwerben, sondern hebt gleichzeitig den Wert aller Programme, die unter Geos ablaufen. Diese sind speziell an die grafische Oberfläche angepaßt, so daß Sie hier immer wieder auf die bereits erwähnten Windows stoßen werden.

Schon im Sonderheft 16 (Einsteiger) haben wir auf den Seiten 56ff. einige Geos-Programme vorgestellt. Die wichtigsten Beispiele wollen wir Ihnen vorstellen.

GeoPaint – das Malprogramm

GeoPaint ist ein ausgezeichnetes Zeichenprogramm, das mit Geos ausgeliefert wird (es befindet sich auf der Systemdiskette).

Seine vielen Bildbearbeitungsfunktionen sind nicht nur sehr bedienerfreundlich zu erreichen (Icons, Pull-down-Menüs), sondern auch außerordentlich schnell in der Ausführung.

Bei einer Auflösung von 320*200 Punkten und 16 Farben sind Grafiken bis zu 640*720 Punkten Größe möglich.

Bild 3 zeigt den Bildschirmaufbau von GeoPaint: In der Mitte befindet sich der jeweils sichtbare Grafikausschnitt, links oben die Menüleiste, am linken Rand die Werkzeugleiste sowie die Musterlupe und rechts unten ein Statusfenster für die unterschiedlichsten Auswahlfunktionen.

Eine besondere GeoPaint-Stärke ist die Einbindung von Texten in die Grafik (»T«-Icon in der Werkzeugleiste), wobei vielfältige Textgestaltungsmöglichkeiten offenstehen. Das Editieren im »Pixel-Modus« (Bildausschnitt wird vergrößert angezeigt) ist ein weiterer Glanzpunkt.

Besonders wichtig ist jedoch die Möglichkeit, GeoPaint-Grafiken in Texte von GeoWrite zu übernehmen.

Der Texteditor GeoWrite

Fast so überzeugend wie GeoPaint ist GeoWrite, der Geos-Texteditor. Er beherrscht die notwendigsten Funktionen eines Texteditors; von einer »richtigen« Textverarbeitung kann man mangels einiger fehlender Möglichkeiten (beispielsweise Suchen/Ersetzen) noch nicht sprechen, doch der separat erhältliche »Writer's Workshop« ist eine Weiterentwicklung ohne Kompromisse.

GeoWrite zeigt jedoch seine Stärken in der Qualität der damit entstanden Textausdrucke; es arbeitet mit Proportionalschrift, verschiedenen Zeichensätzen sowie Schriftarten (kombinierbar) und erlaubt die Übernahme von Grafiken aus GeoPaint in seine Texte.

Auf Dauer ist das Programm allerdings etwas zu unkomfortabel, da die Arbeitsgeschwindigkeit gegenüber einem »normalen« Textverarbeitungsprogramm doch sehr zu wünschen übrig läßt.

Eine Auswahl der unzähligen Schriftarten und -stile zeigt Bild 4.

Hilfsmittel (Desk Accessories)

GeoWrite und GeoPaint sind sogenannte Applikationen, also unter Geos ablaufende Programme. Zusätzlich zu dieser Programmsorte gibt es die Desk Accessories. Diese Programme sind aus jeder Applikation oder dem DeskTop heraus aufrufbar, ohne daß irgendwelche Daten verlorengehen, denn nach der Ausführung eines Desk Accessories wird der alte Zustand wiederhergestellt.

Desk Accessories stehen im »Geos«-Pull-down-Menü zur Verfügung. Ein gutes Beispiel ist der Taschenrechner.

Dieser wird wie ein herkömmlicher Taschenrechner bedient (durch Anklicken der entsprechenden Tasten); mit dem Schließsymbol des Fensters wird er verlassen. Danach wird Ihr GeoWrite-Text oder Ihr GeoPaint-Bild wiederhergestellt.

Der Vorteil von Desk Accessories liegt klar auf der Hand: Sie können bei Bedarf gestartet werden, ohne daß dazu die aktuelle Applikation verlassen und danach neu gestartet werden muß.

Weitere Desk Accessories dienen zum Einstellen einer Alarmzeit (alarm clock), zum Ändern der Voreinstellungen (preference manager) oder zum Festhalten von Notizen (note pad).

Die Zukunft von Geos

Außer der Geos-Software im Lieferumfang wurden zahlreiche weitere Programme entwickelt. Im Sonderheft 16 wurde darauf ausführlich eingegangen.

Da noch mehr Veröffentlichungen geplant sind, steht Geos eine vielseitige und leistungsstarke Produktpalette bevor.

Auch eine Umsetzung auf den C128 (siehe auch 64'er 6/1987) sowie eine deutsche Version sollen nicht mehr lange auf sich warten lassen.

Informationen über Geos

Wo finden Sie weitere Informationen über GEOS. Der Geos-Artikel im 64'er-Sonderheft 16 wurde bereits wiederholt angesprochen. Auch im 64'er-Stammheft nimmt Geos einen fest reservierten Raum ein; von Geos-Erfahrungsberichten über Tips&Tricks bis zu einem eigenen Geos-Kurs findet man dort Monat für Monat die unterschiedlichsten und aktuellsten Informationen.

(Florian Müller/sk/pd)

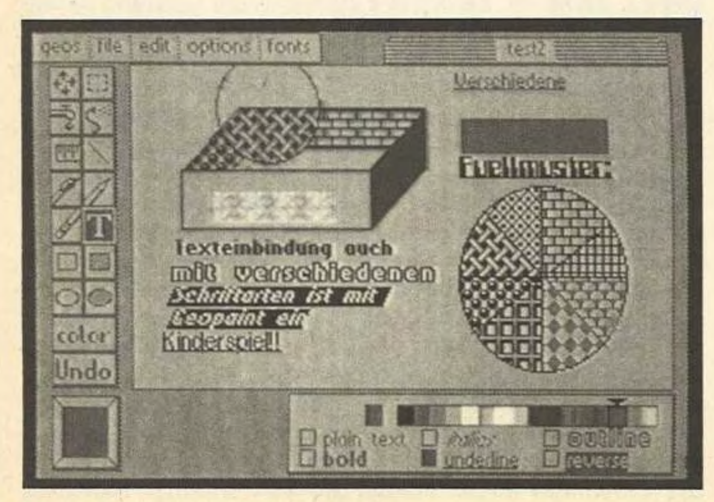


Bild 3. GeoPaint mit Texteinbindung

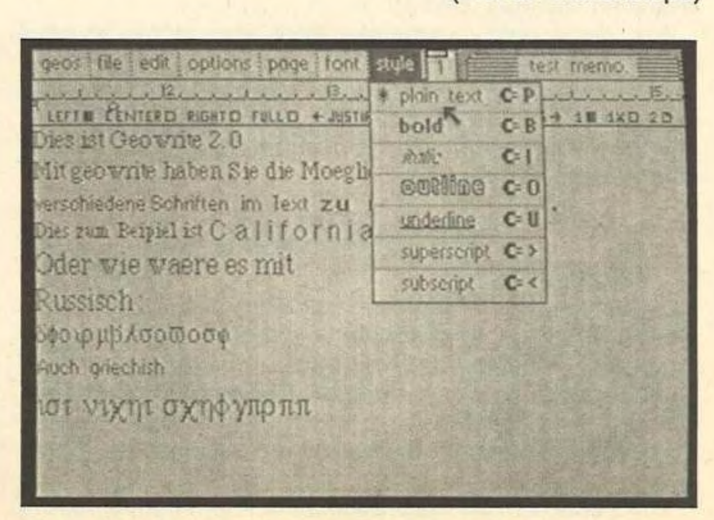


Bild 4. GeoWrite mit einigen seiner Schriftarten

Geschwindigkeit ist Trumpf...

...wenn's um Grafik geht. Wenn Sie dazu noch fünf Bildschirme verwalten und mit mehr als 40 hochwertigen Befehlen und neuen Effekten arbeiten möchten, ist »Hires-Master« genau das richtige für Sie. Hires-Master zählt zu den schnellsten Grafikerweiterungen für den C64.

Da das Basic V2 des C64 leider nicht über Befehle verfügt, mit denen die hochauflösende Grafik ohne POKEs programmiert werden kann, sind einige Basic-Erweiterungen erhältlich, die speziell auf Grafik zugeschnitten sind.

»Hires-Master« ist eine Grafikerweiterung, die sich aber in vielen Beziehungen von anderen Erweiterungen unterscheidet. Hier wurde konsequent Wert auf Geschwindigkeit gelegt. Das heißt, daß die meisten Befehle in bezug auf Schnelligkeit weit über dem Durchschnitt liegen. So setzt der LINE-Befehl zum Beispiel mehr als 13000 Punkte pro Sekunde! Auch Kreise oder ausgefüllte Rechtecke zeichnet das Programm in Windeseile. Hier nun die Leistungsmerkmale von »Hires-Master«:

- Mehr als 30 Befehle für die hochauflösende Grafik
- Sehr komfortabler FILL-Befehl
- Befehle zum Zeichnen sind schneller als in anderen Erweiterungen
- Befehle abkürzbar
- Angabe von Einsprungadressen, dadurch Bedienung auch von Maschinensprache aus möglich
- Bis zu fünf Grafikseiten

Alle 43 Befehle von Hires-Master sind abkürzbar. Die Befehle lassen sich problemlos in bestehende Basic-Programme einbauen.

Nach IF darf das THEN entfallen, so daß Befehle direkt hinter dem Vergleich stehen können.

Hires-Master bietet dem Benutzer vier Grafikseiten, in denen gezeichnet werden kann. Eine fünfte kann zum Zwischenspeichern verwendet werden. Mit diesen Grafikseiten ist verdecktes Zeichnen problemlos möglich. Während in der ersten Grafik gezeichnet wird, sieht der Benutzer eine schon bestehende Grafik. Wenn das Bild fertig ist, wird einfach umgeblendet (sogar flackerfrei möglich).

Zum Zeichnen stehen drei Modi zur Verfügung: Zeichnen, Löschen und Invertieren. Eine Ausnahme bildet dabei der FILL-Befehl, der nur im Modus Zeichnen zu Verfügung steht. Mit ihm ist nur Zeichnen möglich. Das muß aber kein Nachteil sein, wenn man dafür eine nur selten in Grafikerweiterungen verwirklichte Funktion zur Verfügung hat: Das Füllen mit Mustern.

Weil eine Grafik ohne Beschriftung meistens nur eine halbe Sache ist, gibt es einen komfortablen TEXT-Befehl. Mit ihm ist es problemlos möglich, Texte an jede Stelle in die Grafik zu schreiben. Der Text kann sogar in Spiegelschrift erscheinen.

Mit dem CIRCLE-Befehl werden die schnellsten und genauesten Kreise gezeichnet, die auf dem C64 möglich sind. Kreise sind echte Kreise und keine Vielecke. Jeder Punkt hat den kleinstmöglichen Abstand zum Idealkreis (ähnlich der LINE-Routine, wo jeder Punkt den kleinsten Abstand zur Ideallinie hat). Ellipsen werden nicht so schnell gezeichnet wie Kreise, denn der Rechenalgorithmus ist aufwendiger. Das vermindert zwar die Geschwindigkeit, erhöht aber die Genauigkeit.

Schnelle Kreise

Zusätzlich gibt es noch den ARC-Befehl, der nach dem gleichen Muster wie der Ellipsenbefehl arbeitet, mit dem Unterschied, daß sich hier Kreis- oder Ellipsenausschnitte zeichnen lassen.

Es existiert noch ein FCIRCLE-Befehl, mit dem ausgefüllte Kreise in hoher Geschwindigkeit gezeichnet werden.

Zur Veränderung von bestehenden Grafiken gibt es Befehle wie ROLL, SCROLL, DUPLICATE, XMIR und YMIR, um einen Grafikausschnitt zu verschieben, zu kopieren oder an der X- beziehungsweise Y-Achse zu spiegeln.

Besitzer eines MPS 801-Druckers können sich freuen. Erstellte Grafiken lassen sich im Normalformat oder in doppelt großer Ausführung ausdrucken.

Um das Umkopieren von zwei Grafikseiten interessant zu gestalten, gibt es den EFFEKT-Befehl mit 128 Möglichkeiten zum Überblenden. Bildschirme können mit ECLS auf eine ansehnliche Art und Weise gelöscht werden.

Will man zwei Grafikseiten oder Grafik und Text gemeinsam darstellen, kann man den WINDOW-Befehl benutzen. Es brauchen nur zwei Rasterzeilen, an denen umgeschaltet wird, festgelegt werden. Alle Grafikseiten, auch die, die unter dem Betriebssystem liegt ($E000), können gespei-

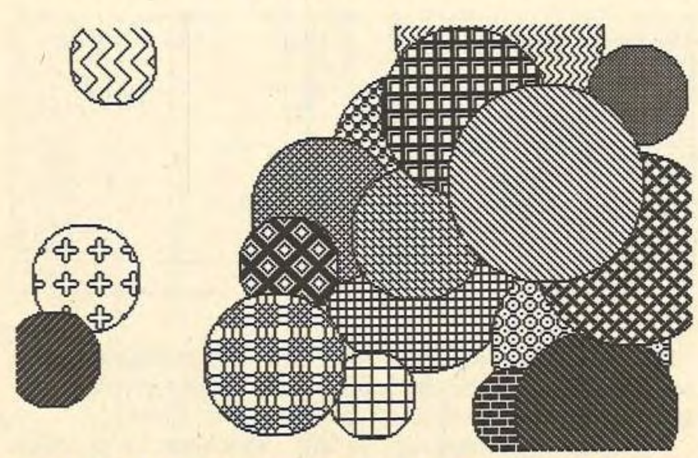


Bild 1. Sie können so viele beliebige Muster darstellen, wie Sie möchten. Der Fill-Mustereditor hilft Ihnen dabei.

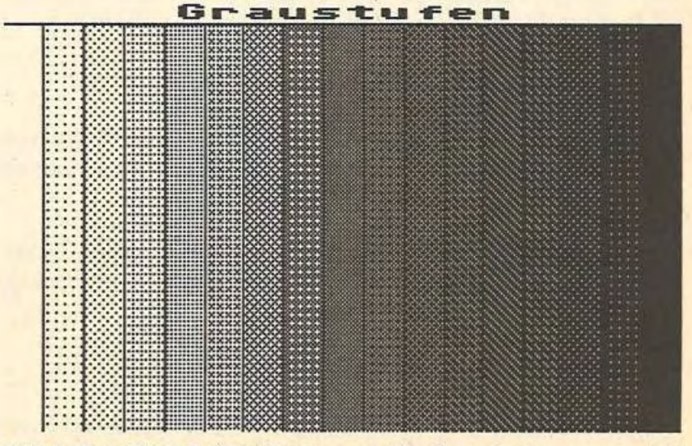


Bild 2. Der FILL-Befehl kann auch für Graustufen verwendet werden. Experimentieren wird hier empfohlen.

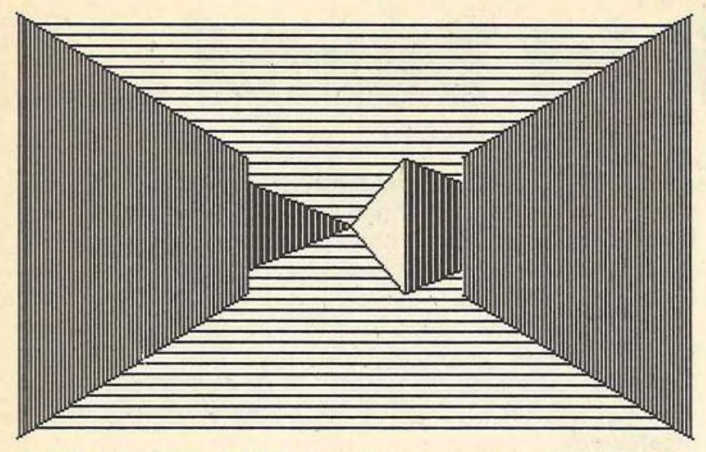


Bild 3. Ein mit dem ROLL-Befehl verschobener Bildschirmausschnitt in dreidimensionaler Darstellung

Bild 4. Texte vielseitig auf den Bildschirm gebracht. So lassen sich Grafiken variantenreich beschriften.

chert und geladen werden. Und das unabhängig davon, ob die Grafik ausgeschaltet ist, da sie weder beim Ein- noch beim Ausschalten gelöscht wird. Am Ende eines Programms oder bei dem Auftreten eines Fehlers wird die Grafik abgeschaltet. Außerdem sorgt eine eigene NMI-Routine dafür, daß selbst bei <RUN/STOP RESTORE> die Grafik ab $E000 nicht zerstört wird. Eine Grafik in diesem Bereich belegt keinen Basic-Speicherplatz.

Befehlsbeschreibung:

Zuvor noch ein paar Hinweise auf Abkürzungen, die bei der Erklärung der Befehle und deren Format benutzt werden:

x: ist die X-Koordinate. Sie darf Werte von 0 bis 319 annehmen.

y: ist die Y-Koordinate. Sie kann zwischen 0 und 199 liegen.

Hier nun die Beschreibung aller in Hires-Master enthaltenen Befehle. In der Klammer hinter dem Befehlswort steht die Abkürzung. Das Zeichen hinter dem Apostroph ist geSHIFTet einzugeben (bei H'E drücken Sie zum Beispiel <H> <SHIFT E>). Die Zeichenbefehle wirken alle auf die aktuelle Grafik, die mit PAGE geändert werden kann.

1. HELP (H'E)
Listet alle verfügbaren Befehlswörter von Hires-Master.

2. INIT (IN'I)
Initialisiert die Erweiterung (setzt Vektoren, Farbe und löscht die Grafik).

3. CLS (-)
Löscht eine Grafik. Alle Bytes dieser Grafik werden auf Null gesetzt.

4. COLOR pf,hf (CO'L)
Setzt die Farben in der Grafik, mit »pf« als Punkt- und »hf« als Hintergrundfarbe. »pf« und »hf« dürfen entsprechend den 16 Farben des C64 nur Werte von 0 bis 15 annehmen.

5. GRON (G'R)
Schaltet die $E000-Grafik ein.

6. GROFF (GRO'F)
Schaltet beliebige Grafik wieder aus und stellt den Zustand her, der vor dem Einschalten mit GRON zu sehen war.

7. MODE m (M'O)
Wechselt den Zeichenmodus. Man kann zwischen Löschen (m=0), Zeichnen (m=1) und Invertieren (m=2) wählen.

8. PLOT x,y (P'L)
Zeichnet einen Punkt in die Grafik an die Stelle x/y.

9. LINE x1,y1,x2,y2 (LI'N)
Zeichnet eine Linie in die aktuelle Grafik. Sollte der Sonderfall einer Horizontal- oder Vertikallinie eintreten, wird zu schnelleren Unterroutinen verzweigt.

10. a:CIRCLE xm,ym,r (C'I)
b:CIRCLE xm,ym,rx,ry (C'I)
Zu a: Zeichnet einen Kreis mit xm/ym als Mittelpunkt und dem Radius r, der Werte zwischen 0 und 255 annehmen darf.
Zu b: Zeichnet Ellipsen mit xm/ym als Mittelpunkt. »rx« und »ry« (0 bis 128) geben die Radien in X- und Y-Richtung an.

11. BLOCK x1,y1,x2,y2 (B'L)
Zeichnet ein ausgefülltes Rechteck mit x1/y1 als linke obere Ecke und x2/y2 als rechte untere Ecke.

12. BOX x1,y1,x2,y2 (B'O)
Zeichnet ein Rechteck mit x1/y1 als linke obere Ecke und x2/y2 als rechte untere Ecke.

13. FILL x,y (F'I)
Füllt einen Bildschirmausschnitt beliebiger Größe und Form mit einem Muster. Der Punkt X/Y muß innerhalb der auszufüllenden Fläche liegen.

14. SETMSK fm,m$ (S'E)
Definiert ein Muster für den FILL-Befehl. Es können bis zu acht Muster intern gespeichert werden. Welches der acht Muster definiert werden soll, läßt sich mit »fm« auswählen. »fm« kann Werte zwischen 0 und 7 annehmen. Der FILL-Befehl benutzt ausschließlich Muster 0 zum Füllen (Bild 1 und 2).

Ein Muster besteht aus 16 Zeilen mit jeweils 16 Punkten. Jede Zeile wird durch vier Hexadezimalzahlen dargestellt. Das heißt: Eine Hex-Zahl legt die Formation von vier Punkten fest. Eine Hex-Zahl entspricht vier Bit oder einer 4stelligen Dualzahl:

Hex-Zahl		Dezimal		Binär	Hex		Dex		Bin
0	-	0	-	0000	8	-	8	-	1000
1	-	1	-	0001	9	-	9	-	1001
2	-	2	-	0010	A	-	10	-	1010
3	-	3	-	0011	B	-	11	-	1011
4	-	4	-	0100	C	-	12	-	1100
5	-	5	-	0101	D	-	13	-	1101
6	-	6	-	0110	E	-	14	-	1110
7	-	7	-	0111	F	-	15	-	1111

Links steht die Hex-Zahl, in der Mitte die Dezimalzahl und rechts das Bit-Muster (Binärzahl). Jede »1« entspricht einem gesetzten, jede »0« einem nicht gesetzten Punkt.

Für das nachfolgende Muster (nächsten Seite, oben links) sind alle Hex-Zahlen mit aufgeführt, darunter findet man ein Beispiel des Befehls:

```
                    1. 2. 3. 4.
0000 0000 0000 0000 0  0  0  0
0000 0011 1100 0000 0  3  C  0
0000 1111 1100 0000 0  F  C  0
0001 1111 1100 0000 1  F  C  0
0011 1100 0011 1111 3  C  3  F
0011 1000 0011 1110 3  8  3  E
0111 0000 0011 1100 7  0  3  C
0111 0000 0000 0000 7  0  0  0
0111 0000 0000 0000 7  0  0  0
0111 0000 0011 1100 7  0  3  C
0011 1000 0011 1110 3  8  3  E
0011 1100 0011 1111 3  C  3  F
0001 1111 1100 0000 1  F  C  0
0000 1111 1100 0000 0  F  C  0
0000 0011 1100 0000 0  3  C  0
0000 0000 0000 0000 0  0  0  0
SETMSK 0,"000003C00FC01FC03C3F383E703C7000
          7000703C383E3C3F1FC00FC003C00000"
```

Der String hat also eine Länge von 64 Zeichen (pro Zeile vier Hex-Zahlen macht bei 16 Zeilen 4 x 16 = 64 Zeichen). Sehen Sie sich das Muster doch mal an:

```
GRON:CLS:CIRCLE160,100,50:FILL160,100:WAIT198,1
```

Der WAIT-Befehl wartet auf eine Taste.

15. CHAMSK ma1,ma2 (CH'A)

Vertauscht die intern gespeicherten Masken miteinander. »ma1« und »ma2« dürfen Werte zwischen 0 und 7 annehmen.

16. STOMSK ma1,ma2 (STO'M)

Kopiert die Maske mit Nummer »ma1« über die Maske Nummer »ma2«.

17. ROLL r,x1,y1,x2,y2 (R'O)

Rollt einen Grafikausschnitt um einen Punkt. »x1/y1« kennzeichnen die linke obere und »x2/y2« die rechte untere Ecke. Der Parameter »r« bestimmt die Richtung:

r=0 – rechts
r=1 – links
r=2 – oben
r=3 – unten

Rollen bedeutet, daß keine Punkte beim Verschieben verlorengehen. Alle aus dem Bildschirm hinausgeschobenen Punkte werden auf der gegenüberliegenden Seite wieder hineingeschoben (Bild 3).

18. SCROLL r,x1,y1,x2,y2 (S'C)

Die Parameter des SCROLL-Befehls entsprechen denen des ROLL-Befehls. Im Unterschied zum ROLL-Befehl gehen alle hinausgeschobenen Punkte verloren.

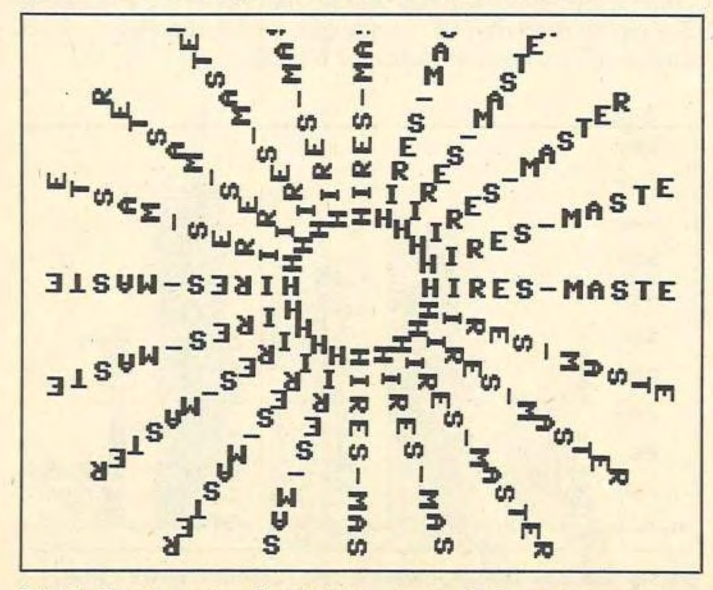


Bild 5. Text kann in alle Richtungen gedreht werden



Bild 6. DUPLICATE kopiert Bildschirmausschnitte in jeder Größe und an jede beliebige Position

19. TEXT x,y,t$ (T'E)

Der Text, der in »t$« steht, wird an Position x/y in die Grafik geschrieben. Im String dürfen folgende Cursor-Steuerzeichen vorkommen:
CLR, HOME, CRSR+UP/DOWN/LEFT/RIGHT, RETURN, CTRL+N (CHR$(14) Kleinschrift), CHR$(142) (Großschrift), RVS ON und RVS OFF.

20. SIZE vx,vy,ax,ay(,d) (SI'Z)

Der Befehl SIZE gibt die Vergrößerung der mit TEXT auszugebenden Zeichen an: »vx« steht für die X-Richtung und »vy« für die Y-Richtung. »ax« und »ay« definieren den Abstand zum nächsten Zeichen. Werden den Variablen ax beziehungsweise ay negative Werte zugewiesen, läßt sich der Text entsprechend in alle vier Richtungen ausgeben. »d« (von 0 bis 3) gibt die Anzahl der 90-Grad-Drehungen eines Zeichens an. Es wird gegen den Uhrzeigersinn gedreht. (Es versteht sich von selbst, daß SIZE vor dem TEXT-Befehl im Programm stehen muß, damit die Parameter richtig wirken) (Bild 4 und 5).

21. ARC xm,ym,rx,ry,sw,ew (A'R)

Mit diesem Befehl können auf einfache Art und Weise Ellipsenausschnitte gezeichnet werden. »xm/ym« geben wieder die Koordinaten des Mittelpunktes an und »rx/ry« die Radien in X- beziehungsweise Y-Richtung. »sw« ist der Startwinkel und »ew« der Endwinkel. Null Grad liegt, wie bei einer Uhr, senkrecht über dem Mittelpunkt. Es wird im Uhrzeigersinn gezeichnet. Wollen Sie zum Beispiel einen 270-Grad-Ausschnitt aus einer Ellipse zeichnen, so muß der Endwinkel um 270 Grad größer sein als der Startwinkel. Beispiel: Eine Ellipse im Mittelpunkt der Grafik:

```
GRON:CLS:MODE1:ARC160,100,90,30,45,315:WAIT198,1
```

22. RAD xm,ym,rx,ry,w (R'A)

Dieser Befehl zeichnet einen Radius in eine imaginäre Ellipse. »xm/ym/rx/ry« wie oben. »w« gibt den Winkel (0° bis 360°) an, unter dem der Radius gezeichnet wird.

23. FCIRCLE xm,ym,r (F'C)

FCIRCLE zeichnet gefüllte Kreise. »xm/ym« definieren wieder den Mittelpunkt, »r« stellt den Radius dar, den der ausgefüllte Kreis haben soll. Es sind Werte zwischen 0 und 255 erlaubt.

24. ECLS verz (E'C)

ECLS löscht die Grafik. »verz« (0 bis 255) gibt die Geschwindigkeit des Löschens an. Der Grafikschirm wird effektvoll gelöscht. Lassen Sie sich überraschen!

25. REVERS (RE'V)

Invertiert eine Grafik. Gesetzte Punkte werden zu gelöschten und umgekehrt.

26. FIGURE x,y,f$ (FI'G)

In »f$« sind die relativen Koordinaten einer Figur gespeichert. Den Zahlen von 1 bis 8 im String kommt die beson-

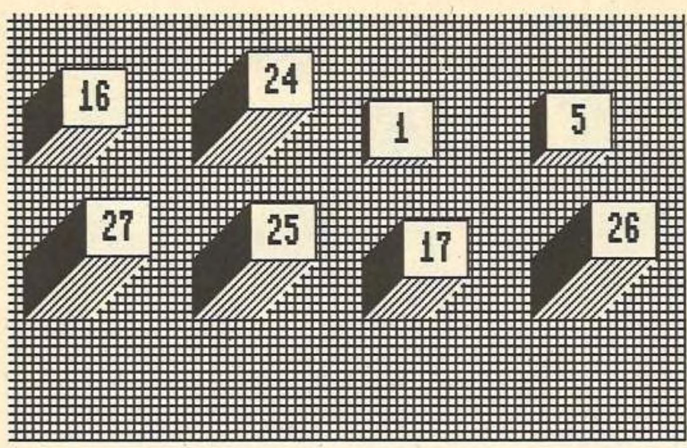


Bild 7. Durch Duplizieren und Verschieben von Bildbereichen lassen sich recht reizvolle Effekte erzielen

dere Bedeutung der Koordinatenspeicherung zu: Eine »1« bedeutet <Schritt nach oben>, »2« <Schritt nach rechts>. Eine »3« bedeutet <Schritt nach unten> und eine »4« <Schritt nach links>. Bei den nachfolgenden Zahlen wird erst ein Punkt gesetzt (im Sinne des Zeichenmodus) und dann ein Schritt in die Richtung gemacht, die die Zahlen angeben. Eine »5« bedeutet <Plotten und Schritt nach oben>, »6«, »7« und »8« jeweils <Plotten und Schritt nach rechts, unten oder links>. Soll zum Beispiel ein Quadrat gezeichnet werden, so muß »f$« so definiert werden:

```
f$="5678"
```

Der Befehl lautet dann:

```
FIGURE 160,100,"5678"
```

oder

```
FIGURE 160,100,f$
```

Aber was sieht man in der Grafik? Einen dicken Punkt. Das Quadrat hat die Ausdehnung 2 an jeder Seite. Man könnte jetzt jede Zahl in f$ vervielfachen, aber es gibt noch einen anderen Weg: Den Befehl TURN.

27. TURN v,d (T'U)

Mit TURN kann die mit FIGURE zu zeichnende Figur vergrößert und gedreht werden. »v« entspricht dem Vergrößerungsfaktor und kann Werte zwischen 0 und 255 annehmen. »d« (0 bis 7) gibt die Anzahl der 45-Grad-Drehungen an. Der Wert 3 dreht die Figur um 135 Grad (3*45=135). Da TURN den Befehl FIGURE beeinflußt, muß TURN vor FIGURE stehen, damit er wirken kann.

28. XMIR x1,y1,x2,y2 (X'M)

Mit dem Befehl XMIR wird der Grafikausschnitt, den x1/y1 und x2/y2 eingrenzen, an der X-Achse gespiegelt (auf den Kopf gestellt).

29. YMIR x1,y1,x2,y2 (Y'M)

Der Befehl ist gleichbedeutend mit XMIR, mit dem Unterschied, daß der Grafikausschnitt an der Y-Achse gespiegelt wird (seitenverkehrt).

30. PAGE p1,p2 (P'A)

Mit dem Befehl PAGE wird festgelegt, in welcher Grafik gezeichnet wird und welche sichtbar ist. »p1« stellt die Grafik ein, in der gezeichnet wird, »p2« die, welche sichtbar ist. »p1« und »p2« können Werte von 0 bis 7 annehmen. Aber nicht jeder Wert entspricht einem anderen Grafikspeicher. Die Werte »0«, »5«, »6« und »7« beschreiben alle den Grafikspeicher ab $E000. Hier eine Tabelle über die PAGE-Werte mit den dadurch eingeschalteten Grafik- und Farbspeichern:

Voreingestellt (nach dem ersten Start) ist PAGE 7,7. Die Grafikseite 4 kann nicht angezeigt werden, da dort der Zeichensatz eingeblendet ist. Diese Grafik wird beim FILL-Befehl zum Zwischenspeichern benutzt.

Wert	Grafikadresse		Farbadresse	
	Hex	Dez	Hex	Dez
0	$E000	57344	$DC00	56320
1	$2000	8192	$0800	2048
2	$4000	16384	$6000	24576
3	$6000	24576	$5C00	23552
(4)	$8000	32768	$8000	32768
5	$E000	57344	$DC00	56320
6	$E000	57344	$DC00	56320
7	$E000	57344	$DC00	56320

31. DUPLICATE x1,y1,x2,y2,x,y(,p1,p2) (D'U)

DUPLICATE kopiert Bildschirmausschnitte. Dabei geben x1/y1 und x2/y2 (links oben und rechts unten) die Eckpunkte eines rechteckigen Ausschnitts an, der so kopiert wird, daß x/y die neue linke obere Ecke angeben. »p1« und »p2« sind PAGE-Parameter. Sie sind optional anzuwenden. Hiermit ist es möglich, einen Grafikausschnitt von einer Grafikseite in eine andere zu kopieren. Es wird von p1 nach p2 kopiert, wobei p1 und p2 die Nummern der beim PAGE-Befehl aufgeführten Grafikspeicher darstellen (Bild 6 und 7).

32. CONNECT p1,p2,mo (CON'N)

CONNECT verknüpft einzelne Grafikseiten miteinander. Es wird p1 mit p2 verknüpft und das Ergebnis nach p2 geschrieben. »mo« gibt die Art der Verknüpfung an:

mo = 0 – Kopieren
mo = 1 – OR-Verknüpfung
mo = 2 – AND-Verknüpfung
mo = 3 – EXOR-Verknüpfung

Dabei bedeutet OR, daß das neue Bild alle gesetzten Punkte beider Grafikseiten enthält. Bei AND werden nur die Punkte übernommen, die auf beiden Seiten gesetzt sind. Etwas komplizierter ist die EXOR-Verknüpfung. Ist auf einer der Grafikseiten ein Punkt gesetzt, wird er im neuen Bild gelöscht, andernfalls wird er gesetzt.

33. SWAP p1,p2,mo (S'W)

Mit dem SWAP-Befehl werden zwei Grafikseiten miteinander verknüpft und gleichzeitig vertauscht. »p1« wird mit »p2« verknüpft, dann p2 nach p1 kopiert und danach das Ergebnis der Verknüpfung nach p2 geschrieben. »mo« hat die gleiche Bedeutung wie bei CONNECT.

34. EFFECT p1,p2,a,verz (E'F)

Mit EFFECT werden zwei Grafikseiten miteinander vertauscht. »a« gibt die Art der Vertauschung an. Es sind Werte von 0 bis 255 erlaubt, wobei bei den geraden Zahlen das gleiche Ergebnis wie bei den nächst höheren ungeraden Zahlen zu erwarten ist. »verz« gibt die Wartezeit an, nachdem ein Byte kopiert wurde (0 bis 255).

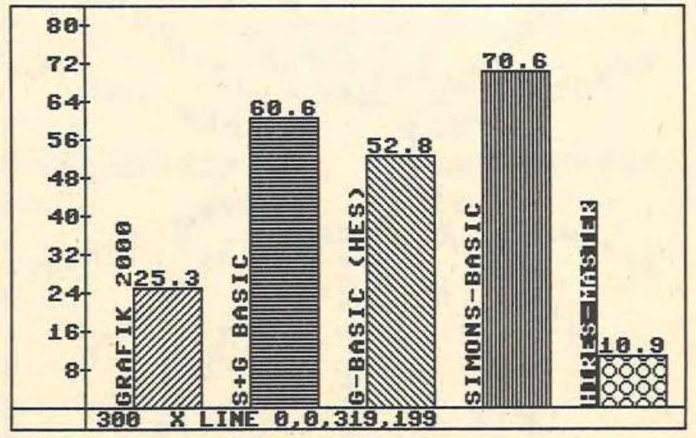


Bild 8. Geschwindigkeitsvergleich: Zeichnen einer Linie. Hires-Master liegt eindeutig vorn. Angaben in Sekunden.

35. GSAVE p,fi$,ga (G'S)
Die Grafik mit der Nummer »p« wird unter dem Namen »fi$« auf dem Gerät mit der Geräteadresse »ga« gespeichert. Die Bildschirmfarbe wird dabei der Rahmenfarbe gleichgesetzt.
36. GLOAD p,fi$,ga (G'L)
Die Grafik mit Namen »fi$« wird vom Gerät mit Geräteadresse »ga« in den Grafikspeicher »p« geladen.
37. WINDOW r1,r2,p1,p2 (W'I)
Mit dem WINDOW-Befehl läßt sich der Bildschirm in zwei Bereiche aufteilen: Von Zeile »r1« bis Zeile »r2« wird die Grafik »p1« dargestellt und von Zeile »r2« bis zum unteren Rand Grafik »p2«. Will man statt zwei Grafiken Text und Grafik darstellen, muß statt der Grafiknummer ein Wert von 128 eingesetzt werden. WINDOW speichert beim Aufruf die aktuellen VIC-Werte und setzt sie dann, wenn der TEXT-Modus angewählt ist, also ein Wert von 128 im WINDOW-Befehl steht.
Deshalb ist es ratsam, den Befehl bei ausgeschalteter Grafik zu aktivieren. Gibt man nur WINDOW ein, wird der Zustand vor dem Einschalten wieder hergestellt.
38. COPY p(,1) (CO'P)
Mit diesem Befehl wird eine Hardcopy im MPS 801-Format an den Drucker gesandt. Der Seikosha GP 100 VC und Epson-Drucker mit zum Beispiel Wiesemann- oder Data-Becker-Interface drucken ebenfalls diese Hardcopies. »p« gibt auch hier die Grafikseite an, die gedruckt werden soll. Zum Ausgeben einer invertierten Grafik muß ein Wert von 128 zum Page-Wert addiert werden. Soll die Hardcopy über eine volle Seite gehen, so wird noch »,1« an den Page-Wert angehängt.

Testfunktionen

39. CSET x,y,pf(,hf) (C'S)
Mit CSET werden gezielt Farben gesetzt. »pf« gibt die Punktfarbe und »hf« die Hintergrundfarbe an. »hf« kann alternativ verwendet werden.
40. OPTION ON/OFF (OP'T)
Nach dem Start von Hires-Master befindet man sich im OPTION ON-Modus.
Alle Hires-Master-Befehle werden beim LISTen invertiert dargestellt. Abschalten kann man das durch OPTION OFF.
41. OFF (O'F)
Schaltet Hires-Master aus und initialisiert alle Vektoren wie bei <RUN/STOP RESTORE>.
42. TEST (x,y) (TE'S)
Der Funktionswert wird »1«, wenn ein Punkt an der Stelle x/y gesetzt war, ansonsten »0«.

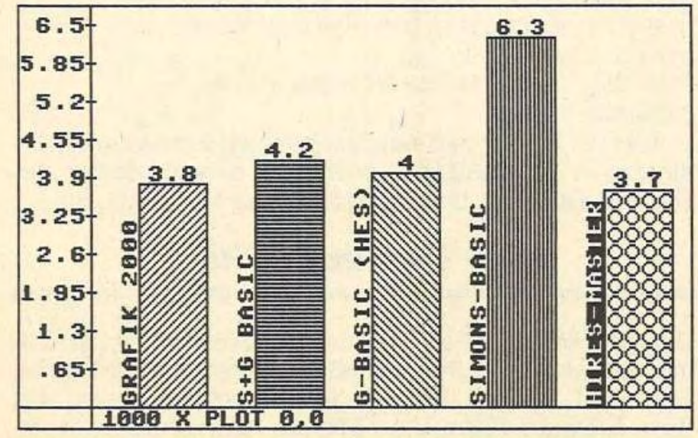


Bild 9. Zeichnen eines Punktes. Hier ist kaum eine höhere Geschwindigkeit zu erreichen.

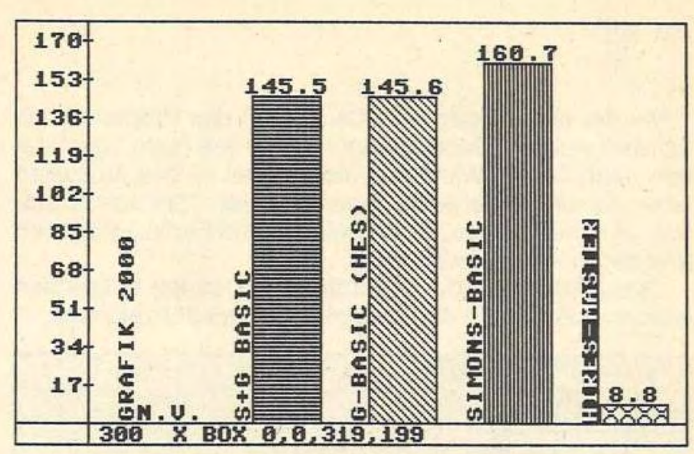


Bild 10. Geschwindigkeitsvergleich: Ein Rahmen wird gezeichnet. Mehr als 15mal schneller: Hires-Master

43. CTEST (x,y,pp) (C'T)
Hiermit wird die aktuelle Farbe an der Stelle x/y getestet. Ob Vorder- oder Hintergrundfarbe wird mit »pp« entschieden:

pp=0 testet die Punktfarbe
pp=1 testet die Hintergrundfarbe.

Das Testergebnis ist immer der Farbcode (eine Zahl zwischen 0 und 15).
Eines der Hauptmerkmale dieser Erweiterung sind die extrem kurzen Ausführungszeiten der einzelnen Zeichenbefehle. Wenn hier nur Zeichenbefehle aufgeführt sind, so liegt das daran, daß nur diese entsprechend optimiert wurden, denn ein Geschwindigkeitsvorteil ergibt sich meistens nur durch Einbuße von Speicherplätzen. Dadurch würde das Programm einfach zu lang.

Vergleiche mit anderen Erweiterungen

Kommen wir zu den eigentlichen Zeiten: Verglichen wurden fünf Basic-Erweiterungen (HSG-W, Grafik 2000, S+G-Basic, G-Basic (von HES) und Simons-Basic). Zur Dokumentation dienen die Bilder 8 bis 13. Es wurden die Zeiten von sieben Befehlen getestet (alle Zeiten sind Sekundenangaben).
Ihnen ist sicherlich aufgefallen, daß Hires-Master bei dem Befehl TEXT zurückliegt. In der G-Basic-Erweiterung von HES-Software gibt es zwar einen TEXT-Befehl, der schneller ist als der von Hires-Master, aber dieser leistet nicht soviel:
So kann er zum Beispiel nur waagerecht in einer Richtung schreiben, keine Buchstaben drehen, und als wichtigsten Punkt: Er hat nur 40 Spalten und 25 Zeilen zum Positionieren des Textes. Soviel zu den Befehlen, deren Ausführungszeiten und Leistungen.

Einsprungadressen
Für alle, die schon etwas von Maschinensprache verstehen, folgt eine Tabelle über die Routine, die der Interpreter anspringt, um einen Befehl oder eine Funktion auszuführen.
Alle Aufrufe werden mit ausgeschaltetem Interrupt (SEI), ausgeschalteten ROMs und I/O ausgeführt.
Also:

```
SEI
LDA #$30
STA $01
JSR $XXXX  ;BEFEHLS- oder FUNKTIONSADRESSE
LDA #$37
```

```
STA $01
CLI
RTS
```

Vor der erstmaligen Benutzung muß das Programm initialisiert werden. Dazu springt man so wie oben beschrieben nach $A001. Wichtig in diesem Teil ist das Aufstellen einer Tabelle, welche die Adressen jeder Grafikzeile enthält. Hier nun die einzelnen Befehls- und Funktionsnamen und deren Adressen im RAM.

Spezialroutinen, die von anderen Routinen aufgerufen werden, sind durch ein Sternchen (*) gekennzeichnet.

Adresse	Funktion	Adresse	Funktion
$A001	- Initialisieren <*>	$B3FF	- SCROLL
$A08D	- CLS	$B678	- DUPLICATE
$A0F5	- COLOR	$B881	- RAD
$A124	- GRON	$B9A6	- REVERS
$A152	- GROFF	$B9C0	- GSAVE
$A173	- PLOT	$BA27	- GLOAD
$A1BD	- LINE	$BA40	- CONNECT
$A3DC	- HLINE <*LINE>	$BA9D	- SWAP
$A4A6	- VLINE <*LINE>	$BB00	- FIGURE
$A504	- CIRCLE	$BB72	- TURN
$A613	- BLOCK	$BB94	- PAGE
$A71D	- BOX	$BC15	- EFFECT
$A741	- ELLIPSE <*CIRCLE>	$BCA8	- WINDOW
$A814	- FILL	$BD93	- XMIR
$AA79	- SETMSK	$BE84	- YMIR
$AAB2	- CHAMSK	$BF24	- OPTION
$AAD9	- STOMSK	$BF37	- FUNK. TEST
$AAF8	- ROLL	$C000	- INIT (Befehl) <*>
$AAE9	- TEXT	$C20F	- Neuer IRQ
$B012	- SIZE	$C253	- COPY
$B061	- HELP	$C41B	- CSET
$B087	- ECLS	$C47F	- FUNK. CTEST
$B109	- ARC	$C4A3	- Subroutinen <*>
$B318	- FCIRCLE	$C797	- Tabellen (bis $CB14)

Die Adressenangaben beziehen sich auf den Anfang der Befehle. Es müssen also noch Parameter geholt werden. Dazu wird in den Programmteil »Subroutinen« verzweigt, um die entsprechenden Parameter zu holen. Die Adresse $A001 wird vom INIT-Befehl aus angesprungen.

Hier werden alle nötigen Vektoren und Speicherzellen gesetzt. HLINE ab $A3DC zeichnet horizontale, VLINE ab $A4A6 vertikale Linien. Es wird erwartet, daß sich die Koordinaten in den entsprechenden Speicherzellen befinden.

Bei den meisten Befehlen ist mindestens die Angabe des Koordinatenpaares x/y erforderlich (PLOT, LINE, CIRCLE etc.), beim LINE-Befehl sogar zwei Paare. Die Speicherzel-

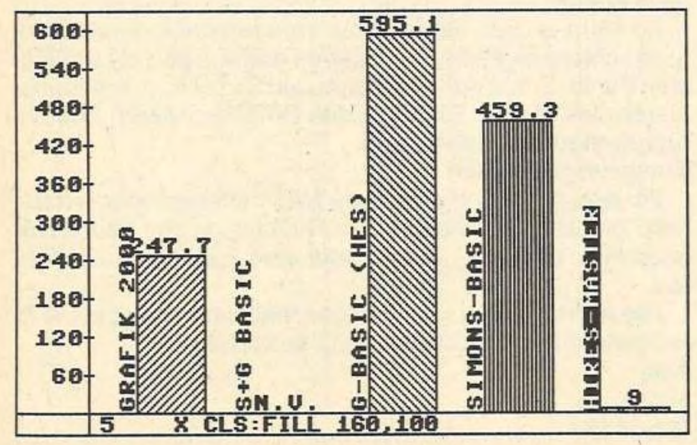


Bild 11. Geschwindigkeitsvergleich: Füllen mit einem Muster. Auch hier eine extrem überdurchschnittliche Leistung.

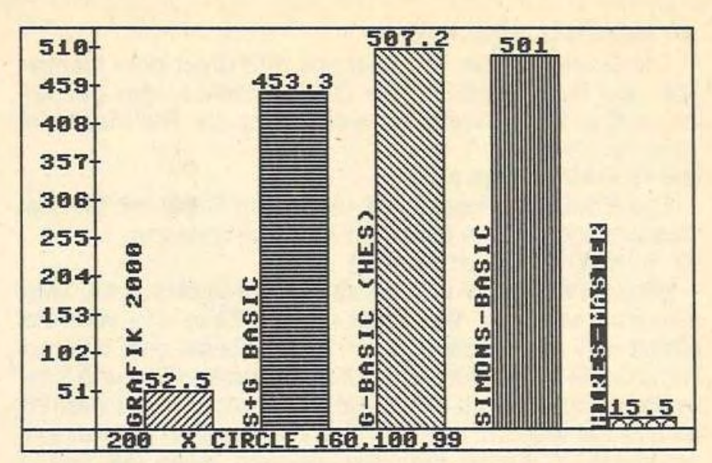


Bild 12. Zeichnen eines Kreises. Hires-Master ist etwa 30mal schneller als die meisten Konkurrenten.

len für diese Koordinaten: (Die X-Koordinate braucht immer zwei Speicherzellen, da bei ihr Werte von 0 bis 319 erlaubt sind)

1. Koordinatenpaar (x/y oder x1/y1):
   X-Koordinate:$AC/AD
   Y-Koordinate:$AE
2. Koordinatenpaar (x2/y2):
   X-Koordinate:$C1/C2
   Y-Koordinate:$02

Soll zum Beispiel ein Kreis mit dem Radius 66 in die Mitte der Grafik gesetzt werden, so sieht man sich den Programmteil ab $A504 an und sucht die Stelle, ab der alle Parameter vom Basic-Interpreter geholt werden (diese Stelle muß übersprungen werden, da man ja von Maschinensprache aus einen Kreis zeichnen möchte). Der besagte Einsprung liegt bei $A519. Der Radius muß sich vor dem Aufruf im X-Register befinden, die Koordinaten des Mittelpunktes in $AC/$AD (X-Koordinate) und $AE (Y-Koordinate).

Das Programm (es liegt im Kassettenpuffer!) würde nun so lauten:

```
033C SEI        ;INTERRUPT AUS
033D LDA #$30   ;SPEICHER KOMPLETT AUF RAM
033F STA $01
0341 JSR $A001  ;TABELLE INITIALISIEREN
0344 LDA #$A0   ;160 LOW-BYTE DER X-KOORDINATE
0346 LDY #$00   ;0 HIGH-BYTE DER X-KOORDINATE
0348 LDX #$64   ;100 Y-KOORDINATE
034A STA $AC    ;KOORDINATEN SPEICHERN
034C STY $AD
034E STX $AE
0350 LDX #$42   ;66 (RADIUS)
0352 JSR $A519  ;EINSPRUNG ZUR KREISROUTINE
0355 LDA #$37   ;ALTE RAM-ROM-KONFIGURATION
0357 STA $01
0359 CLI        ;INTERRUPT WIEDER EIN
035A RTS
```

ACHTUNG! Bei den meisten Befehlen werden die Koordinaten in ($AC/$AD,$AE) verändert, deshalb sollte man sie, falls sie weiter benötigt werden, vorher speichern.

Der Fill-Mustereditor

Der recht komfortable Editor zum Erstellen von neuen Füllmustern wird mit RUN gestartet. Nun baut sich die Arbeitsfläche auf. Das mit Punkten gefüllte Rechteck entspricht dem späteren Füllmuster. Mit den Cursor-Tasten kann beliebig darin herumgefahren werden. Mit der Leer-Taste wird der Punkt, an dem sich der Cursor momentan befin-

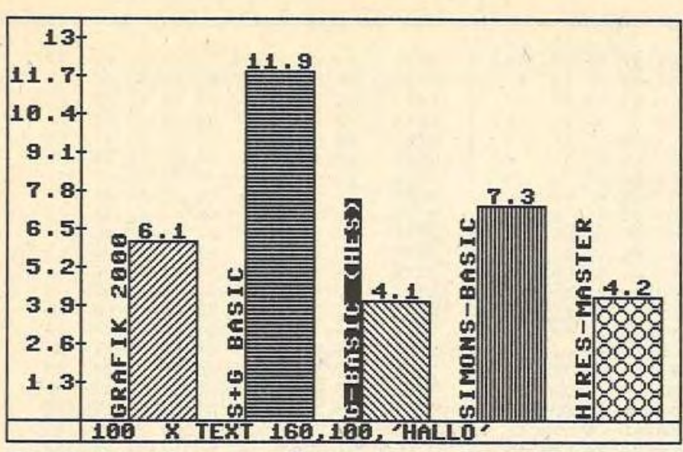


Bild 13. Schreiben eines Textes. Hier ist Hires-Master wegen seiner komplizierten Berechnungen an zweiter Stelle.

det, invertiert, das heißt, ist er gelöscht, so wird er gesetzt und umgekehrt.

Um ein möglichst großes Arbeitstempo zu gewährleisten, wurde auf allzu große Übersichtlichkeit des Programms verzichtet.

Man muß nun erst einmal <B> drücken, um in den »Befehls-Eingabe«-Modus zu gelangen. Jetzt kann man durch Betätigen der entsprechenden Taste einen von 17 Befehlen aussuchen:

- <O> rechnet das Arbeitsfeld in Hexadezimalzahlen um – entsprechend den Anforderungen der Grafikerweiterungen. Diese können mit dem Befehl »Setmsk« in die Füllmaske umgesetzt werden.
- <E> beendet das Programm. Hierzu ist außer dem Hinweis, daß das Muster dann verloren ist, nichts mehr zu sagen.
- <,> veranlaßt den Cursor an den Anfang der Zeile zu springen.
- <.> Cursor springt an das Ende der Zeile.
- <X> hält die momentane X-Position des Cursors fest. Sie wird über dem Arbeitsfeld mit einem Strich gekennzeichnet.
- <A> läßt den Cursor in die vorher mit <X> gekennzeichnete Spalte springen.
- <Y> arbeitet analog zu dem <X>-Befehl, allerdings merkt sich der Computer nun die Y-Position.
- <B> bringt den Cursor in die entsprechende Zeile.
- <Z> löscht die gesamte Zeile, in der sich der Cursor gerade aufhält.
- <S> löscht die momentane Spalte.
- <SHIFT Z> setzt die ganze Zeile, ebenso wie
- <SHIFT S> die ganze Spalte füllt.
- <CBM Z> invertiert die Zeile.
- <CBM S> invertiert die Spalte.
- <Home> läßt den Cursor in die linke obere Ecke springen, während
- <CLR> noch den Bildschirm löscht, bevor der Cursor nach links oben springt.
- <I> invertiert das gesamte Arbeitsfeld.

Der Fill-Mustereditor ist wesentlich komfortabler in der Handhabung als der eingebaute SETMSK-Befehl.

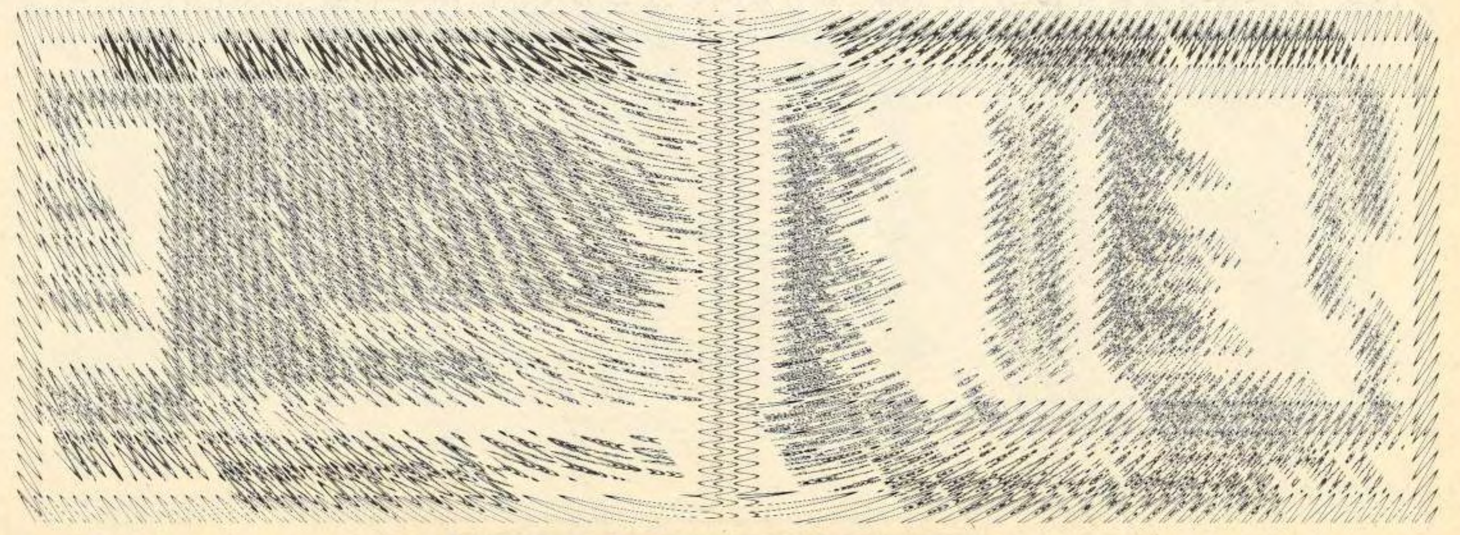
Eingabehinweise

Geben Sie bitte die Grafikerweiterung »Hires-Master« (Listing 1) mit dem MSE ein und speichern diese. Der Start erfolgt mit RUN. Das Listing 2 ist für die Funktion nicht erforderlich, es dient nur als Beispiel für die Leistungsfähigkeit von Hires-Master. Listing 3, den Fill-Mustereditor, geben Sie bitte mit dem Checksummer ein. Es benötigt zum Betrieb den Hires-Master.

(J. Schwarzer/S. Thelen/og)


```
Name : hires-master           0801 3418
-----------------------------------------
0801 : 4d 08 c2 07 9e 28 32 31   3a
0809 : 34 34 29 20 0e 05 22 c8   c9
0811 : 49 52 45 53 2d cd 41 53   2c
0819 : 54 45 52 20 49 53 20 57   07
0821 : 52 49 54 54 45 4e 20 20   3f
0829 : 20 20 20 20 20 20 20 20   29
0831 : 20 20 20 20 20 20 20 20   31
0839 : 20 42 59 20 ca 45 53 4b   8f
0841 : 4f 20 d3 43 48 57 41 52   e7
0849 : 5a 45 52 00 00 00 44 30   4c
0851 : 00 00 00 00 00 00 00 00   52
0859 : 00 00 00 00 00 00 00 a9   ad
0861 : 00 a0 a0 a2 09 85 f7 85   d5
0869 : f9 84 fa 86 f8 a2 2c a8   da
0871 : b1 f7 91 f9 c8 d0 f9 e6   8a
0879 : f8 e6 fa ca d0 f2 4c 00   d2
0881 : c0 c0 00 00 00 00 00 00   a2
0889 : 00 00 00 00 00 00 00 d6   37
0891 : 45 52 53 49 4f 4e 20 32   4a
0899 : 00 30 20 4b 4f 4d 4d 54   60
08a1 : 20 42 41 4c 44 2e 20 d6   a0
08a9 : 49 45 4c 4c 45 49 43 48   6e
08b1 : 54 20 4d 49 54 20 d2 4f   c2
08b9 : 54 41 54 45 20 55 4e 44   da
08c1 : 20 cd 41 47 4e 49 46 59   fc
08c9 : 2e 00 00 00 00 00 00 00   f8
08d1 : 00 00 00 00 00 00 00 00   d2
08d9 : 00 00 00 00 00 00 00 00   da
08e1 : 00 00 00 00 00 00 00 00   e2
08e9 : 00 00 00 00 00 00 00 00   ea
08f1 : 00 00 00 00 00 00 00 00   f2
08f9 : 00 00 00 00 00 00 00 00   fa
0901 : a9 ed a0 c1 8d 00 03 8c   ff
0909 : 01 03 a9 fe a0 c1 8d 02   28
```

Listing 1. »Hires-Master« – bitte mit dem MSE eingeben



```
0911 : 03 8c 03 03 a9 1a a0 c0   eb
0919 : 8d 04 03 8c 05 03 a9 56   b6
0921 : a0 c1 8d 06 03 8c 07 03   7d
0929 : a9 d2 a0 c0 8d 08 03 8c   ba
0931 : 09 03 a9 26 a0 c1 8d 0a   4d
0939 : 03 8c 0b 03 20 d6 c1 a9   b9
0941 : 00 a0 80 a2 e0 85 f7 84   29
0949 : f8 86 fa aa 18 a0 00 98   50
0951 : 65 f7 9d 00 d8 9d 00 da   49
0959 : 08 a5 f8 69 00 9d 00 d9   40
0961 : 28 a5 fa 69 00 9d 00 db   ec
0969 : e8 c8 c0 08 90 e1 a5 f7   85
0971 : 69 3f 85 f7 08 a5 f8 69   3e
0979 : 01 85 f8 28 a5 fa 69 01   5a
0981 : 85 fa e0 c8 90 c7 20 8d   b7
0989 : a0 4c 0f a1 a0 fa a9 00   d0
0991 : 99 ff df 99 f9 e0 99 f3   4a
0999 : e1 99 ed e2 99 e7 e3 99   ba
09a1 : e1 e4 99 db e5 99 d5 e6   26
09a9 : 99 cf e7 99 c9 e8 99 c3   29
09b1 : e9 99 bd ea 99 b7 eb 99   6e
09b9 : b1 ec 99 ab ed 99 a5 ee   dc
09c1 : 99 9f ef 99 99 f0 99 93   07
09c9 : f1 99 8d f2 99 87 f3 99   21
09d1 : 81 f4 99 7b f5 99 75 f6   92
09d9 : 99 6f f7 99 69 f8 99 63   e6
09e1 : f9 99 5d fa 99 57 fb 99   d5
09e9 : 51 fc 99 4b fd 99 45 fe   48
09f1 : 88 d0 9d 60 20 3d c6 8a   71
09f9 : 0a 0a 0a 0a 8d 97 c7 20   c1
0a01 : 46 c6 20 3d c6 8a 29 0f   de
0a09 : 0d 97 c7 8d 97 c7 ad 97   23
0a11 : c7 a0 fa 99 ff db 99 f9   53
0a19 : dc 99 f3 dd 99 ed de 88   10
0a21 : d0 f1 60 ad ab c7 d0 28   44
0a29 : ee ab c7 a9 35 85 01 ad   f3
0a31 : 00 dd 8d a4 c7 a9 00 8d   fd
0a39 : 00 dd ad 18 d0 8d a2 c7   2a
0a41 : a9 78 8d 18 d0 ad 11 d0   ed
0a49 : 8d a3 c7 a9 3b 8d 11 d0   d5
0a51 : 60 ad ab c7 f0 1b a9 35   65
0a59 : 85 01 ad a4 c7 8d 00 dd   03
0a61 : ad a2 c7 8d 18 d0 ad a3   09
0a69 : c7 8d 11 d0 a9 00 8d ab   7d
0a71 : c7 60 a6 ae a5 ad c9 01   d9
0a79 : d0 04 a5 ac c9 40 b0 31   0e
0a81 : a5 ac 90 09 20 bf c5 85   e4
0a89 : ac 84 ad a0 00 18 29 f8   4e
0a91 : 7d 00 da 85 f7 a5 ad 7d   d4
0a99 : 00 db 85 f8 a5 ac 29 07   7a
0aa1 : aa bd d9 c7 2c 98 c7 70   21
0aa9 : 0e 30 07 49 ff 31 f7 91   47
0ab1 : f7 60 51 f7 91 f7 60 11   a8
0ab9 : f7 91 f7 60 20 21 c6 a4   f2
0ac1 : ae c4 02 d0 03 4c dc a3   b9
0ac9 : a6 ad a5 ac e4 c2 d0 07   fa
0ad1 : c5 c1 d0 23 4c a6 a4 e4   66
0ad9 : c2 f0 1c b0 1e 38 a5 c1   ee
0ae1 : e5 ac 85 fb a5 c2 e5 ad   60
0ae9 : 85 fc a9 34 a2 a3 8d 1d   95
0af1 : a3 8e 1e a3 4c 0f a2 c5   2b
0af9 : c1 90 e2 e5 c1 85 fb 8a   c5
0b01 : e5 c2 85 fc a9 55 a2 a3   5f
0b09 : 8d 1d a3 8e 1e a3 c4 02   f6
0b11 : b0 14 38 a5 02 e5 ae 85   a3
0b19 : ba a9 76 a0 a3 8d 04 a3   57
0b21 : 8c 05 a3 4c 36 a2 98 e5   49
0b29 : 02 85 ba a9 a8 a0 a3 8d   0b
0b31 : 04 a3 8c 05 a3 a4 fb a5   65
0b39 : fc d0 04 c4 ba 90 0a 4a   24
0b41 : 98 6a 85 5c 85 fd 4c 6d   50
0b49 : a2 a5 ba 85 fb 4a 85 5c   fe
0b51 : 85 fd 84 ba ad 04 a3 ac   30
0b59 : 1d a3 8d 1d a3 8c 04 a3   45
0b61 : ad 05 a3 ac 1e a3 8d 1e   81
0b69 : a3 8c 05 a3 a0 00 84 5d   df
0b71 : a6 02 18 a5 c1 29 f8 7d   17
0b79 : 00 da 85 f9 a5 c2 7d 00   ed
0b81 : db 85 fa a5 c1 29 07 aa   69
0b89 : bd d9 c7 85 bf 2c 98 c7   25
0b91 : 70 0c 30 07 49 ff 31 f9   42
0b99 : 4c a1 a2 51 f9 2c 11 f9   c2
0ba1 : 91 f9 20 9a c6 20 a5 a1   d1
0ba9 : a5 ba 8d f5 a2 a5 fb 8d   30
0bb1 : 0a a3 8d 19 a3 a5 fc 8d   8a
0bb9 : fe a2 8d 10 a3 2c 98 c7   fb
0bc1 : 70 12 30 08 a2 31 a0 21   c0
0bc9 : a9 ff d0 0e a2 51 a0 41   21
0bd1 : a9 00 f0 06 a2 11 a0 01   ae
0bd9 : a9 00 8d 22 a3 8e 23 a3   ac
0be1 : 8d 2a a3 8c 2b a3 a6 fd   64
0be9 : f0 48 a4 f7 a2 00 86 f7   5a
0bf1 : 18 a5 5c 69 00 85 5c 8a   d3
0bf9 : 65 5d 85 5d c9 00 f0 15   a4
0c01 : 90 19 20 ff ff 38 a5 5c   37
0c09 : e9 00 85 5c a5 5d e9 00   cc
0c11 : 85 5d 4c 1c a3 a5 5c c9   48
0c19 : 00 b0 e7 20 ff ff a5 be   83
0c21 : 49 00 11 f7 91 f7 a5 bf   9c
0c29 : 49 00 01 f9 81 f9 c6 fd   f1
0c31 : d0 be 60 46 be 90 0a 66   a7
0c39 : be 98 69 08 a8 90 02 e6   84
0c41 : f8 06 bf b0 01 60 26 bf   6d
0c49 : a5 f9 e9 07 85 f9 90 01   b2
0c51 : 60 c6 fa 60 06 be 90 0a   8c
0c59 : 26 be 98 e9 07 a8 b0 02   be
0c61 : c6 f8 46 bf b0 01 60 66   8e
0c69 : bf a5 f9 69 08 85 f9 b0   9c
0c71 : 01 60 e6 fa 60 98 29 07   39
0c79 : c9 07 f0 0a c8 a5 f9 29   37
0c81 : 07 f0 16 c6 f9 60 98 69   37
0c89 : 38 a8 a5 f8 69 01 85 f8   44
0c91 : a5 f9 29 07 f0 03 c6 f9   94
0c99 : 60 38 a5 f9 e9 39 85 f9   30
0ca1 : a5 fa e9 01 85 fa 60 98   41
0ca9 : 29 07 f0 0c 88 a5 f9 29   03
0cb1 : 07 c9 07 f0 19 e6 f9 60   ee
0cb9 : 38 98 e9 39 a8 a5 f8 e9   4e
0cc1 : 01 85 f8 a5 f9 29 07 c9   10
0cc9 : 07 f0 03 e6 f9 60 a5 f9   13
0cd1 : 69 38 85 f9 a5 fa 69 01   d1
0cd9 : 85 fa 60 a5 c2 a6 c1 c5   9c
0ce1 : ad d0 02 e4 ac b0 0c a4   dd
0ce9 : ad 85 ad 84 c2 a4 ac 86   66
0cf1 : ac 84 c1 a5 ac 48 a5 ad   03
0cf9 : 48 a5 c1 85 ac a5 c2 85   43
0d01 : ad a0 00 20 9a c6 86 af   5c
0d09 : a5 f7 85 f9 a5 f8 85 fa   78
0d11 : 68 85 ad 68 85 ac 20 9a   28
0d19 : c6 a5 f7 c5 f9 d0 11 a5   1e
0d21 : f8 e5 fa d0 0b bd e1 c7   9a
0d29 : a6 af 3d e9 c7 4c a5 a1   ec
0d31 : bd e1 c7 48 38 a5 f9 e5   3e
0d39 : f7 aa a5 fa e5 f8 4a 8a   b2
0d41 : 6a 4a 4a aa 8c 69 a4 2c   b7
0d49 : 98 c7 70 0f 30 38 ce 69   96
0d51 : a4 8c 6e a4 a9 31 8d 6a   9b
0d59 : a4 d0 0a a9 ff 8d 6e a4   8c
0d61 : a9 11 8d 6a a4 18 68 49   83
0d69 : 00 11 f7 2c a9 ff 91 f7   46
0d71 : a5 f7 69 08 85 f7 90 03   ce
0d79 : e6 f8 18 ca d0 ee a6 af   b9
0d81 : bd e9 c7 4c a5 a1 18 68   47
0d89 : 2c a9 ff 51 f7 91 f7 a5   eb
0d91 : f7 69 08 85 f7 90 03 e6   cd
0d99 : f8 18 ca d0 ec [illegible] [illegible] bd   a8
0da1 : e9 c7 4c a5 a1 a5 [illegible] c5   10
0da9 : ae b0 08 a4 ae 85 ae 84   21
0db1 : 02 38 98 e5 ae 48 a5 ae   d3
0db9 : 29 07 a8 a5 ae 29 f8 aa   b2
0dc1 : 20 9c c6 68 aa e8 ad 98   c8
0dc9 : c7 d0 06 a5 be 49 ff 85   70
0dd1 : be a5 be 2c 98 c7 70 0c   39
0dd9 : 30 05 31 f7 4c e7 a4 51   10
0de1 : f7 4c e7 a4 11 f7 91 f7   94
0de9 : ca f0 17 c8 c0 08 90 e1   5d
0df1 : a5 f7 69 3f 85 f7 a5 f8   75
0df9 : 69 01 85 f8 a5 be a0 00   36
0e01 : f0 cf 60 20 b5 c5 20 46   8c
0e09 : c6 20 3d c6 86 b6 20 79   99
0e11 : 00 c9 2c d0 03 4c 41 a7   02
0e19 : a0 00 84 b7 86 fb 84 fd   28
0e21 : 84 fe a5 ac a6 ad 85 c1   95
0e29 : 86 c2 a5 ae 85 02 18 a5   64
0e31 : c1 65 fb 85 ac a5 c2 69   2a
0e39 : 00 85 ad 18 a5 02 65 b7   da
0e41 : b0 09 c9 c8 b0 05 aa 20   1f
0e49 : 75 a1 38 a5 02 e5 b7 b0   e1
0e51 : 10 38 a5 c1 e5 fb 85 ac   cc
0e59 : a5 c2 e9 00 85 ad 4c 78   c2
0e61 : a5 aa 85 ae 20 75 a1 38   37
0e69 : a5 c1 e5 fb 85 ac a5 c2   c1
0e71 : e9 00 85 ad 20 73 a1 18   c6
0e79 : a5 02 65 b7 b0 08 c9 c8   74
0e81 : b0 04 aa 20 75 a1 18 a5   f2
0e89 : c1 65 b7 85 ac a5 c2 69   71
0e91 : 00 85 ad 18 a5 02 65 fb   ba
0e99 : b0 09 c9 c8 b0 05 aa 20   77
0ea1 : 75 a1 38 a5 02 e5 fb b0   4a
0ea9 : 10 38 a5 c1 e5 b7 85 ac   02
0eb1 : a5 c2 e9 00 85 ad 4c d0   ca
0eb9 : a5 aa 85 ae 20 75 a1 38   8f
0ec1 : a5 c1 e5 b7 85 ac a5 c2   91
0ec9 : e9 00 85 ad 20 73 a1 18   1e
0ed1 : a5 02 65 fb b0 08 c9 c8   54
0ed9 : b0 04 aa 20 75 a1 20 e7   ef
0ee1 : a5 f0 a3 4c 2f a5 18 a5   3d
0ee9 : b7 65 fd 85 fd 90 02 e6   bd
0ef1 : fe 38 e5 fb aa a5 fe e9   ac
0ef9 : 00 e6 b7 c5 fe d0 02 e4   5b
0f01 : fd b0 06 c6 fb 86 fd 85   a8
0f09 : fe a5 fb c5 b7 b0 02 68   6b
0f11 : 68 60 20 21 c6 20 69 c6   76
0f19 : a5 ae a4 02 84 ae 85 02   57
0f21 : 18 a5 c1 29 f8 79 00 da   b2
0f29 : 85 f9 a5 c2 79 00 db 85   7f
0f31 : fa a5 c1 29 07 aa bd e9   24
0f39 : c7 85 af 20 9a c6 a0 00   15
0f41 : a5 f7 c5 f9 d0 1f a5 f8   21
0f49 : e5 fa d0 19 bd e1 c7 25   57
0f51 : af 85 b6 a5 b6 20 a5 a1   6b
0f59 : a5 ae c5 02 f0 06 20 02   cb
0f61 : c5 4c 54 a6 60 bd e1 c7   41
0f69 : 85 b8 38 a5 f9 e5 f7 aa   11
0f71 : a5 fa e5 f8 4a 8a 6a 4a   63
0f79 : 4a 85 b6 a5 f7 85 f9 a5   c7
0f81 : f8 85 fa 8c ab a6 2c 98   5e
0f89 : c7 70 0f 30 54 ce ab a6   0a
0f91 : 8c b0 a6 a9 31 8d ac a6   d4
0f99 : d0 0a a9 ff 8d b0 a6 a9   25
0fa1 : 11 8d ac a6 18 a5 b8 a6   58
0fa9 : b6 49 00 11 f7 2c a9 ff   ae
0fb1 : 91 f7 a5 f7 69 08 85 f7   83
0fb9 : 90 03 e6 f8 18 ca d0 ee   9c
0fc1 : a5 af 20 a5 a1 a5 ae c5   88
0fc9 : 02 f0 99 a5 f9 85 f7 a5   55
0fd1 : fa 85 f8 20 02 c5 a5 f7   a5
0fd9 : 85 f9 a5 f8 85 fa 4c a5   90
0fe1 : a6 18 a5 b8 a6 b6 2c a9   38
0fe9 : ff 51 f7 91 f7 a5 f7 69   20
0ff1 : 08 85 f7 90 03 e6 f8 18   47
0ff9 : ca d0 ec a5 af 20 a5 a1   f1
1001 : a5 ae c5 02 f0 15 a5 f9   f1
1009 : 85 f7 a5 fa 85 f8 20 02   f7
1011 : c5 a5 f7 85 f9 a5 f8 85   13
1019 : fa d0 c6 60 20 21 c6 20   9f
1021 : 69 c6 20 f4 a3 a5 ae 48   47
1029 : 20 a6 a4 a5 02 85 ae 20   c1
1031 : f4 a3 68 85 ae a5 c1 a6   2e
1039 : c2 85 ac 86 ad 4c a6 a4   db
1041 : 8a 48 20 73 00 20 3d c6   e9
1049 : 68 85 b6 e4 b6 d0 03 4c   55
1051 : 19 a5 e0 81 90 02 a2 80   4a
1059 : 86 b7 a6 b6 e0 81 90 02   9c
1061 : a2 80 86 b6 a0 ff 84 bb   4f
1069 : 84 bc c8 84 b8 a5 ac 85   84
1071 : c1 a5 ad 85 c2 a5 ae 85   40
1079 : 02 a6 b8 bd 76 c8 a6 b6   6a
1081 : 20 01 a8 85 ba a5 b8 49   4b
1089 : ff aa bd 76 c8 a6 b7 20   fc
1091 : 01 a8 85 b9 c5 bb d0 06   08
1099 : a5 ba c5 bc f0 44 18 a5   81
10a1 : c1 65 ba 85 ac a5 c2 69   4a
10a9 : 00 85 ad 38 a5 02 e5 b9   54
10b1 : 90 05 aa 20 75 a1 18 a5   83
10b9 : 02 65 b9 b0 32 c9 c8 b0   e8
10c1 : 2e aa 85 ae 20 75 a1 38   20
10c9 : a5 c1 e5 ba 85 ac a5 c2   f9
10d1 : e9 00 85 ad 20 73 a1 38   66
10d9 : a5 02 e5 b9 90 04 aa 20   44
10e1 : 75 a1 a5 ba 85 bc a5 b9   30
10e9 : 85 bb e6 b8 d0 8b 60 38   78
10f1 : a5 c1 e5 ba 85 ac a5 c2   21
10f9 : e9 00 85 ad 30 e4 10 d7   13
1101 : 85 f7 8e 0e a8 a2 07 98   d4
1109 : 46 f7 90 02 69 00 6a ca   85
1111 : d0 f6 60 20 b5 c5 a0 00   84
1119 : ae 00 db a9 80 84 f7 84   08
1121 : f9 86 f8 85 fa a2 20 b1   f5
1129 : f7 91 f9 c8 d0 f9 e6 f8   eb
1131 : e6 fa ca d0 f2 84 b7 20   d3
1139 : b8 c6 31 f7 f0 01 60 84   41
1141 : b8 84 b9 a5 ae f0 0c 20   41
1149 : 02 c5 b1 f7 25 be f0 f3   8d
1151 : 20 1d c5 20 b8 c6 b1 f7   ee
1159 : f0 03 4c a0 a9 a9 ff 91   fd
1161 : f7 a5 ae 29 0f 85 f9 a5   4c
1169 : ac 29 08 0a 65 f9 aa 18   ee
1171 : a5 f7 85 f9 a5 f8 69 60   3b
1179 : 85 fa bd f4 ca 91 f9 a5   f6
1181 : ad d0 06 a5 ac c9 08 90   27
1189 : 48 38 a5 f7 e9 08 85 f9   3e
1191 : a5 f8 e9 00 85 fa b1 f9   18
1199 : d0 09 a5 b8 d0 33 e6 b8   22
11a1 : 4c ad a8 4a b0 29 a5 b8   94
11a9 : d0 27 e6 b8 a6 b7 a5 ac   f6
11b1 : 29 07 85 f9 18 a5 ac e5   2c
11b9 : f9 9d 00 d5 a5 ad e9 00   ab
11c1 : 9d 00 d6 a5 ae 9d 00 d7   50
11c9 : e0 ff f0 05 e6 b7 2c 84   6b
11d1 : b8 a5 ad f0 06 a5 ac c9   b9
11d9 : 38 b0 44 18 a5 f7 69 08   4d
11e1 : 85 f9 a5 f8 69 00 85 fa   8e
11e9 : b1 f9 d0 09 a5 b9 d0 2f   b6
11f1 : e6 b9 4c ff a8 0a b0 25   af
11f9 : a5 b9 d0 23 e6 b9 a6 b7   5a
1201 : 18 a5 ac 09 07 69 01 9d   33
1209 : 00 d5 a5 ad 69 00 9d 00   20
1211 : d6 a5 ae 9d 00 d7 e0 ff   5b
1219 : f0 05 e6 b7 2c 84 b9 20   4a
1221 : 1d c5 a5 ae c9 c8 b0 07   14
1229 : b1 f7 d0 25 4c 5e a8 c6   96
1231 : b7 a6 b7 e0 ff d0 01 60   90
1239 : bd 00 d5 85 ac bd 00 d6   83
```

```
1241 : 85 ad bd 00 d7 85 ae 20  b1
1249 : b8 c6 31 f7 d0 e1 4c 40  7d
1251 : a8 c9 ff f0 da aa a5 ac  ef
1259 : 29 f8 85 ac 84 af 8a 39  58
1261 : d9 c7 d0 35 a5 af d0 1d  4e
1269 : e6 af 8a 48 a6 b7 a5 ac  eb
1271 : 9d 00 d5 a5 ad 9d 00 d6  ae
1279 : a5 ae 9d 00 d7 e0 ff f0  43
1281 : 02 e6 b7 68 aa e6 ac c8  17
1289 : c0 08 90 d2 a5 ac d0 02  d3
1291 : c6 ad c6 ac a0 00 4c 30  11
1299 : a9 a9 00 85 af f0 e6 84  ef
12a1 : b8 84 b9 20 b8 c6 11 f7  04
12a9 : 91 f7 18 a5 f7 85 f9 a5  cf
12b1 : f8 69 60 85 fa a5 ae 29  10
12b9 : 0f 85 fb a5 ac 29 08 0a  87
12c1 : 05 fb aa bd f4 ca 25 be  de
12c9 : 11 f9 91 f9 a5 ac 05 ad  a9
12d1 : f0 41 a5 be 0a 90 36 a5  ec
12d9 : f7 e9 08 85 f9 a5 f8 e9  fc
12e1 : 00 85 fa b1 f9 4a b0 29  a0
12e9 : a5 b8 d0 27 e6 b8 a6 b7  42
12f1 : 38 a5 ac e9 01 9d 00 d5  0d
12f9 : a5 ad e9 00 9d 00 d6 a5  70
1301 : ae 9d 00 d7 e0 ff f0 0b  61
1309 : e6 b7 4c 14 aa 31 f7 f0  56
1311 : d7 84 b8 a5 ad f0 06 a5  d3
1319 : ac c9 3f b0 41 a5 be 4a  61
1321 : 90 36 a5 f7 69 07 85 f9  0d
1329 : a5 f8 69 00 85 fa b1 f9  90
1331 : 0a b0 29 a5 b9 d0 27 e6  1f
1339 : b9 a6 b7 18 a5 ac 69 01  9e
1341 : 9d 00 d5 a5 ad 69 00 9d  6a
1349 : 00 d6 a5 ae 9d 00 d7 e0  ee
1351 : ff f0 0b e6 b7 4c 5f aa  19
1359 : 31 f7 f0 d7 84 b9 20 1d  8e
1361 : c5 a5 ae c9 c8 b0 0e b1  8b
1369 : f7 d0 03 4c 40 a8 25 be  6e
1371 : d0 03 4c a4 a9 4c 30 a9  7c
1379 : 20 eb c4 85 af 20 46 c6  13
1381 : 20 31 c6 c0 40 f0 03 4c  34
1389 : 48 b2 a0 00 b1 22 aa c8  bb
1391 : b1 22 20 d9 c4 a6 af 9d  12
1399 : f4 ca c8 b1 22 aa c8 b1  59
13a1 : 22 20 d9 c4 a6 af 9d 04  49
13a9 : cb e6 af c8 c0 40 90 dc  f6
13b1 : 60 20 eb c4 85 af 69 20  70
13b9 : 85 be 20 46 c6 20 eb c4  15
13c1 : aa a4 af bd f4 ca 48 b9  9b
13c9 : f4 ca 9d f4 ca 68 99 f4  68
13d1 : ca e8 c8 c4 be 90 ec 60  bf
13d9 : 20 eb c4 85 af 69 20 85  a2
13e1 : be 20 46 c6 20 eb c4 a8  df
13e9 : a6 af bd f4 ca 99 f4 ca  58
13f1 : e8 c8 e4 be 90 f4 60 20  c1
13f9 : 3d c6 8a 48 20 46 c6 68  65
1401 : d0 03 4c a9 ab c9 01 f0  8a
1409 : 11 c9 02 d0 03 4c f9 ac  6d
1411 : c9 03 d0 03 4c 3d ac 4c  ea
1419 : 1b c6 20 ac c4 a5 c1 29  08
1421 : 07 85 b9 aa bd e9 c7 85  04
1429 : 5f 49 ff 85 15 bd d9 c7  14
1431 : 85 bc 20 9a c6 86 b8 bd  6f
1439 : e1 c7 85 5e 49 ff 85 14  fe
1441 : bd d9 c7 85 bb a5 f7 85  61
1449 : f9 a5 f8 85 fa a5 af d0  41
1451 : 07 a5 5e 25 5f 4c 82 ab  21
1459 : a5 14 31 f7 85 ba b1 f7  38
1461 : 25 5e 4a 05 ba 91 f7 a6  4e
1469 : af 08 18 a5 f9 69 08 85  ed
1471 : f9 90 02 e6 fa 28 ca f0  0e
1479 : 08 b1 f9 6a 91 f9 4c 6a  14
1481 : ab b1 f9 aa 25 15 85 ba  5f
1489 : 8a 25 bc f0 06 a5 bb 11  92
1491 : f7 91 f7 8a 6a 25 5f 05  f7
1499 : ba 91 f9 a5 ae c5 02 f0  52
14a1 : 06 20 02 c5 4c 46 ab 60  57
14a9 : 20 ac c4 a5 ac 29 07 85  40
14b1 : b9 aa bd e1 c7 85 5f 49  24
14b9 : ff 85 15 bd d9 c7 85 bc  e3
14c1 : a5 c1 a6 c2 85 ac 86 ad  7c
14c9 : 20 9a c6 86 b8 bd e9 c7  69
14d1 : 85 5e 49 ff 85 14 bd d9  7b
14d9 : c7 85 bb a5 f7 a6 f8 85  aa
14e1 : f9 86 fa a6 af d0 07 a5  9a
14e9 : 5e 25 5f 4c 16 ac b1 f9  bd
14f1 : 25 14 85 ba b1 f9 25 5e  15
14f9 : 0a 05 ba 91 f9 08 38 a5  73
1501 : f9 e9 08 85 f9 b0 02 c6  5c
1509 : fa 28 ca f0 08 b1 f9 2a  32
1511 : 91 f9 4c fe ab b1 f9 aa  17
1519 : 25 15 85 ba 25 bc f0 28  28
1521 : 06 a5 bb 11 f7 91 f7 8a  0c
1529 : 2a 25 5f 05 ba 91 f9 a5  ca
1531 : ae c5 02 f0 06 20 02 c5  55
1539 : 4c dc ab 60 20 ac c4 38  d5
1541 : a5 ae e5 02 f0 40 48 20  69
1549 : bc ad a5 af d0 39 a5 b8  1a
```

```
1551 : 25 ba 85 bc 49 ff 85 bd  f2
1559 : 68 aa b1 f7 48 a5 f7 85  1e
1561 : f9 a5 f8 85 fa 20 02 c5  60
1569 : b1 f7 25 bc 85 b6 b1 f9  bf
1571 : 25 bd 05 b6 91 f9 ca d0  43
1579 : e4 68 25 bc 85 b6 b1 f7  37
1581 : 25 bd 05 b6 91 f7 60 68  c8
1589 : 48 aa a5 f7 85 fb a5 f8  4f
1591 : 85 fc b1 f7 48 a5 f8 85  a0
1599 : fa a5 f7 85 f9 29 07 d0  bb
15a1 : 0f 38 a5 f7 e9 39 85 f7  a3
15a9 : a5 f8 e9 01 85 f8 d0 02  cc
15b1 : c6 f7 b1 f7 25 b8 85 b6  7a
15b9 : b1 f9 25 b9 05 b6 91 f9  28
15c1 : ca d0 d2 68 25 b8 85 b6  51
15c9 : b1 f7 25 b9 05 b6 91 f7  33
15d1 : 18 a5 fb 69 08 85 f7 a5  c0
15d9 : c 69 00 85 f8 c6 af 30  1f
15e1 : 15 f0 08 a9 ff 85 b8 84  be
15e9 : b9 d0 9c a5 ba 85 b8 a5  ec
15f1 : bb 85 b9 4c 88 ac 68 60  b7
15f9 : 20 ac c4 20 69 c6 38 a5  9d
1601 : 02 e5 ae f0 42 48 a0 00  a8
1609 : 20 bc ad a5 af d0 39 a5  59
1611 : b8 25 ba 85 bc 49 ff 85  dc
1619 : bd 68 aa b1 f7 48 a5 f8  35
1621 : 85 fa a5 f7 85 f9 20 1f  72
1629 : c5 b1 f7 25 bc 85 b6 b1  a0
1631 : f9 25 bd 05 b6 91 f9 ca  42
1639 : d0 e4 68 25 bc 85 b6 b1  70
1641 : f7 25 bd 05 b6 91 f7 60  73
1649 : 68 48 aa a5 f7 85 fb a5  1b
1651 : f8 85 fc b1 f7 48 a5 f8  cb
1659 : 85 fa a5 f7 85 f9 29 07  9e
1661 : c9 07 d0 0f a5 f7 69 38  f4
1669 : 85 f7 a5 f8 69 01 85 f8  19
1671 : 4c 76 ad e6 f7 b1 f7 25  78
1679 : b8 85 b6 b1 f9 25 b9 05  91
1681 : b6 91 f9 ca d0 d0 68 25  57
1689 : b8 85 b6 b1 f7 25 b9 05  81
1691 : b6 91 f7 18 a5 fb 69 08  01
1699 : 85 f7 a5 fc 69 00 85 f8  c1
16a1 : c6 af 30 15 f0 08 a9 ff  e4
16a9 : 85 b8 84 b9 d0 9a a5 ba  d0
16b1 : 85 b8 a5 bb 85 b9 4c 49  5d
16b9 : ad 68 60 a5 ac 48 a5 ad  66
16c1 : 48 a5 c1 85 ac a5 c2 85  0b
16c9 : ad 20 9a c6 bd [illegible] [illegible] 85  5b
16d1 : ba 49 ff 85 bb 68 [illegible] ad  51
16d9 : 68 85 ac 20 9a c6 bd e1  ce
16e1 : c7 85 b8 49 ff 85 b9 60  96
16e9 : 20 b5 c5 20 46 c6 20 31  d7
16f1 : c6 84 b7 a9 d0 8d 99 c7  8c
16f9 : a0 00 84 b8 84 c7 84 b9  dd
1701 : b1 22 30 58 c9 20 b0 47  c9
1709 : c9 0d d0 15 a9 00 85 ac  3a
1711 : 85 ad 18 a5 ae 69 08 85  df
1719 : 33 c9 c8 b0 2f 85 ae 90  74
1721 : 2b c9 0e d0 06 a9 08 85  a7
1729 : b8 d0 21 c9 11 f0 e3 c9  87
1731 : 12 d0 06 a9 80 85 c7 d0  57
1739 : 13 c9 13 f0 32 c9 1d d0  9b
1741 : 0b 18 a5 ac 69 08 85 ac  9d
1749 : 90 02 e6 ad 4c 06 b0 c9  95
1751 : 60 90 04 29 df d0 5c 29  68
1759 : 3f 4c b4 ae 29 7f c9 7f  76
1761 : d0 02 a9 5e c9 20 b0 49  5b
1769 : c9 13 d0 0d 20 8d ad b9  d6
1771 : 00 85 ac 85 ad 85 ae f0  b3
1779 : d3 c9 0d f0 8f c9 0e d0  b3
1781 : 06 a9 00 85 b8 f0 c5 c9  ca
1789 : 11 d0 0b 38 a5 ae e9 08  54
1791 : 90 ba 85 ae b0 b6 c9 12  c1
1799 : d0 06 a9 00 85 c7 f0 ac  8a
17a1 : c9 1d d0 a8 38 a5 ac e9  79
17a9 : 08 85 ac b0 9f c6 ad 90  bd
17b1 : 9b 09 40 05 c7 85 fb a9  6d
17b9 : 00 06 fb 2a 06 fb 2a 06  f5
17c1 : fb 2a 0d 99 c7 65 b8 85  dd
17c9 : fc 78 a9 31 85 01 ae ac  06
17d1 : c7 d0 03 4c 59 af ca d0  6b
17d9 : 2c ca f0 52 a0 07 b1 fb  f2
17e1 : 4a 2e a1 c7 4a 2e a0 c7  cc
17e9 : 4a 2e 9f c7 4a 2e 9e c7  4b
17f1 : 4a 2e 9d c7 4a 2e 9c c7  cb
17f9 : 4a 2e 9b c7 4a 2e 9a c7  4a
1801 : 88 10 db 30 5d a0 07 b1  e9
1809 : fb 4a 6e 9a c7 4a 6e 9b  d8
1811 : c7 4a 6e 9c c7 4a 6e 9d  f0
1819 : c7 4a 6e 9e c7 4a 6e 9f  3c
1821 : c7 4a 6e a0 c7 4a 6e a1  89
1829 : c7 88 10 db 30 34 a0 07  e9
1831 : b1 fb 0a 7e 9a c7 0a 7e  3f
1839 : 9a c7 0a 7e 9a c7 0a 7e  16
1841 : 9a c7 0a 7e 9a c7 0a 7e  1e
1849 : 9a c7 0a 7e 9a c7 0a 7e  26
1851 : 9a c7 e8 88 10 da 30 0a  c7
1859 : a0 07 b1 fb 99 9a c7 88  07
```

```
1861 : 10 f8 20 8e c6 a0 00 84  42
1869 : bb 20 9a c6 a5 ac a6 ad  69
1871 : 85 c1 86 c2 a5 ae 85 02  bb
1879 : a2 00 a9 01 85 b6 a6 bb  c6
1881 : bd 9a c7 85 bc a9 08 85  72
1889 : ba a9 01 85 bd 06 bc 90  29
1891 : 1c a5 ad f0 06 a5 ac c9  dd
1899 : 40 b0 27 a5 be 20 a5 a1  77
18a1 : 20 55 c5 c6 bd d0 ea c6  52
18a9 : ba d0 de f0 15 a5 ad f0  b8
18b1 : 06 a5 ac c9 40 b0 0b 20  e4
18b9 : 55 c5 c6 bd d0 ef c6 ba  77
18c1 : d0 c7 a5 c1 a6 c2 85 ac  06
18c9 : 86 ad e6 ae a6 ae e0 c8  aa
18d1 : b0 11 20 9c c6 c6 b6 a5  6e
18d9 : b6 d0 a3 e6 bb a6 bb e0  5f
18e1 : 08 90 97 a5 02 69 ff 85  42
18e9 : ae 18 a5 ac 69 08 85 ac  e8
18f1 : aa a5 ad 69 00 85 ad c9  7d
18f9 : 01 d0 02 e0 40 b0 11 a5  18
1901 : ae c9 c8 b0 0b e6 b9 a4  f4
1909 : b9 c4 b7 b0 03 4c 01 ae  1c
1911 : 60 20 3d c6 8e 8b af 8a  c3
1919 : 0a 0a 0a 8d 46 ae 8d a9  c0
1921 : ae 20 46 c6 20 3d c6 8e  6e
1929 : 7c af 8a 0a 0a 0a 8d 17  b6
1931 : ae 8d 90 ae 20 46 c6 20  2f
1939 : 85 c5 8d ee af 8c f5 af  79
1941 : 20 46 c6 20 85 c5 8d e7  c6
1949 : af ce e7 af 20 79 00 c9  b0
1951 : 2c d0 0c 20 73 00 20 3d  1f
1959 : c6 8a 29 03 8d ac c7 60  2d
1961 : a9 d5 a0 c9 85 f7 84 f8  72
1969 : a0 00 a2 0a b1 f7 08 29  40
1971 : 7f 20 6e c5 c8 ca 28 f0  ba
1979 : 0c 10 f1 a9 20 20 6e c5  87
1981 : ca d0 fa f0 e5 60 20 3d  ec
1989 : c6 86 b6 86 b9 a0 07 a5  19
1991 : b6 85 b9 a2 fa 1e ff df  2d
1999 : 1e f9 e0 1e f3 e1 1e ed  52
19a1 : e2 1e e7 e3 1e e1 e4 1e  c9
19a9 : db e5 1e d5 e6 1e cf e7  27
19b1 : 1e c9 e8 1e c3 e9 1e bd  31
19b9 : ea 1e b7 eb 1e b1 ec 1e  7d
19c1 : ab ed 1e a5 ee 1e 9f ef  dd
19c9 : 1e 99 f0 1e 93 f1 1e 8d  10
19d1 : f2 1e 87 f3 1e 81 f4 1e  31
19d9 : 7b f5 1e 75 f6 1e 6f f7  93
19e1 : 1e 69 f8 1e 63 f9 1e 5d  ef
19e9 : fa 1e 57 fb 1e 51 fc 1e  e4
19f1 : 4b fd 1e 45 fe ca d0 9d  30
19f9 : a5 b6 f0 31 08 c6 b8 d0  fc  a6
1a01 : c6 b9 d0 f8 88 10 88 60  e3
1a09 : 20 b5 c5 20 46 c6 20 3d  0f
1a11 : c6 e0 81 90 02 a2 80 86  fe
1a19 : b6 20 46 c6 20 3d c6 e0  13
1a21 : 81 90 02 a2 80 86 b7 20  1b
1a29 : 46 c6 20 96 c5 e0 01 d0  b6
1a31 : 02 c0 69 90 04 a2 01 a0  9a
1a39 : 68 84 f9 86 fa 20 46 c6  8a
1a41 : 20 96 c5 e0 01 d0 02 c0  5a
1a49 : 69 90 04 a2 01 a0 68 84  0f
1a51 : fb 86 fc e4 fa d0 02 c4  33
1a59 : f9 b0 0c a5 f9 84 f9 85  19
1a61 : fb a5 fa 86 fa 85 fc a0  cf
1a69 : 00 84 fd 84 fe 84 5e 84  52
1a71 : 5f 88 84 5c 84 5d 84 14  2e
1a79 : 84 15 a5 ac a6 ad 85 c1  f8
1a81 : 86 c2 a5 ae 85 02 a5 f9  9a
1a89 : a6 fa d0 1e c9 5a b0 09  e8
1a91 : aa bd 76 c9 85 fd 4c cb  02
1a99 : b1 c9 b4 b0 0d e9 59 aa  4d
1aa1 : bd 76 c9 49 ff 85 5c 4c  6b
1aa9 : e0 b1 e0 01 d0 02 c9 0e  1a
1ab1 : b0 0b e9 b3 aa bd 76 c9  de
1ab9 : 85 5e 4c fb b1 e9 0e aa  f8
1ac1 : bd 76 c9 49 ff 85 14 4c  6a
1ac9 : 14 b2 a5 fb a6 fc d0 0d  cf
1ad1 : c9 5a b0 09 aa bd 76 c9  1b
1ad9 : 85 fe 4c 21 b2 84 fe a5  ab
1ae1 : fb a6 fc d0 11 c9 b4 b0  1c
1ae9 : 0d e9 59 aa bd 76 c9 49  e0
1af1 : ff 85 5d 4c 21 b2 84 5d  08
1af9 : e6 5d a6 fc a5 fb e0 01  97
1b01 : d0 02 c9 0e b0 0b e9 b3  79
1b09 : aa bd 76 c9 85 5f 4c 21  2f
1b11 : b2 84 5f 38 a5 fb e9 0e  e2
1b19 : aa bd 76 c9 49 ff 85 15  4d
1b21 : 84 bb 84 bc c8 84 b8 a6  1c
1b29 : b8 bd 76 c8 a6 b6 20 01  19
1b31 : a8 85 ba a5 b8 49 ff a6  2a
1b39 : bd 76 c8 a6 b7 20 01 a8  0a
1b41 : 85 b9 c5 bb d0 09 a5 ba  ed
1b49 : c5 bc d0 03 4c 08 b3 a6  22
1b51 : b8 e4 fd 90 26 e4 fe f0  74
1b59 : 02 b0 20 a5 fd 05 fe f0  56
1b61 : 18 a5 c1 65 ba 85 ac     c4
```

Listing 1. (Fortsetzung)

```
1b69 : a5 c2 69 00 85 ad 38 a5  bc
1b71 : 02 e5 b9 90 06 aa 20 75  07
1b79 : a1 a6 b8 e4 5c f0 06 b0  ff
1b81 : 2c e4 5d 90 28 a5 5c c5  35
1b89 : 5d d0 04 c9 ff f0 1e 18  b9
1b91 : a5 c1 65 ba 85 ac a5 c2  a1
1b99 : 69 00 85 ad 18 a5 02 65  9b
1ba1 : b9 b0 0a c9 c8 b0 06 aa  ee
1ba9 : 20 75 a1 a6 b8 e4 5e 90  0e
1bb1 : 2a e4 5f f0 02 b0 24 a5  c5
1bb9 : 5e 05 5f f0 1e 38 a5 c1  4d
1bc1 : e5 ba 85 ac a5 c2 e9 00  12
1bc9 : 85 ad 18 a5 02 65 b9 b0  73
1bd1 : 0a c9 c8 b0 06 aa 20 75  29
1bd9 : a1 a6 b8 e4 15 90 28 e4  d8
1be1 : 14 f0 02 b0 22 a5 14 c5  2f
1be9 : 15 d0 04 c9 ff f0 18 38  f9
1bf1 : a5 c1 e5 ba 85 ac a5 c2  21
1bf9 : e9 00 85 ad 38 a5 02 e5  7e
1c01 : b9 90 04 aa 20 75 a1 a5  d8
1c09 : ba 85 bc a5 b9 85 bb e6  ee
1c11 : b8 f0 03 4c 28 b2 60 20  66
1c19 : b5 c5 20 46 c6 20 3d c6  72
1c21 : a0 00 84 b7 84 bc 84 fd  16
1c29 : 84 fe 86 fb a5 ac a6 ad  03
1c31 : 85 5e 86 5f a5 ae 85 bd  d4
1c39 : 38 a5 5e e5 fb 85 ac a5  82
1c41 : 5f e9 00 85 ad 18 a5 5e  35
1c49 : 65 fb 85 c1 a5 5f 69 00  40
1c51 : 85 c2 38 a5 bd e5 b7 90  05
1c59 : 07 85 ae 85 b9 20 de b3  ff
1c61 : 18 a5 bd 65 b7 b0 16 c9  55
1c69 : c8 b0 12 85 ae 85 ba 2c  19
1c71 : 98 c7 70 06 10 04 c5 bd  7d
1c79 : f0 03 20 de b3 38 a5 5e  1f
1c81 : e5 b7 85 ac a5 5f e9 00  36
1c89 : 85 ad 18 a5 5e 65 b7 85  9b
1c91 : c1 a5 5f 69 00 85 c2 38  d2
1c99 : a5 bd e5 fb 90 16 85 ae  43
1ca1 : c5 b9 f0 10 2c 98 c7 70  08
1ca9 : 08 10 06 c5 bb f0 05 85  56
1cb1 : bb 20 de b3 18 a5 bd 65  1b
1cb9 : fb b0 1a c9 c8 b0 16 85  42
1cc1 : ae c5 ba f0 10 2c 98 c7  73
1cc9 : 70 08 10 06 c5 bc f0 05  12
1cd1 : 85 bc 20 de b3 20 e7 a5  bf
1cd9 : f0 a3 4c 39 b3 a5 ad 10  14
1ce1 : 04 84 ac 84 ad a5 c2 f0  d8
1ce9 : 12 c9 02 b0 06 a5 c1 c9  9f
1cf1 : 40 90 08 a9 3f 85 c1 a9  2b
1cf9 : 01 85 c2 4c f4 a3 20 3d  5e
1d01 : c6 8a 29 03 48 20 46 c6  e3
1d09 : 68 d0 03 4c 9c b4 c9 01  bc
1d11 : f0 0a c9 02 d0 03 4c c9  a3
1d19 : b5 4c 21 b5 20 ac c4 a5  b9
1d21 : c1 29 07 aa bd e9 c7 85  e3
1d29 : 5f 49 ff 85 15 20 9a c6  28
1d31 : bd e1 c7 85 5e 49 ff 85  bd
1d39 : 14 a5 af d0 0d a5 5e 4a  32
1d41 : 25 5f 85 5f a5 14 05 15  9c
1d49 : 85 15 a5 f7 85 f9 a5 f8  72
1d51 : 85 fa a5 af f0 29 a5 14  ca
1d59 : 31 f7 85 ba b1 f7 25 5e  6b
1d61 : 4a 05 ba 91 f7 a6 af 08  92
1d69 : 18 a5 f9 69 08 85 f9 90  b5
1d71 : 02 e6 fa 28 ca f0 08 b1  62
1d79 : f9 6a 91 f9 4c 68 b4 b1  89
1d81 : f9 aa 25 15 85 ba 8a 6a  e8
1d89 : 25 5f 05 ba 91 f9 a5 ae  d3
1d91 : c5 02 f0 06 20 02 c5 4c  16
1d99 : 4b b4 60 20 ac c4 a5 ac  3b
1da1 : 29 07 aa bd e1 c7 85 5f  e1
1da9 : 49 ff 85 15 a5 c1 a6 c2  7e
1db1 : 85 ac 86 ad 20 9a c6 bd  51
1db9 : e9 c7 85 5e 49 ff 85 14  86
1dc1 : a6 af d0 0d a5 5e 0a 25  d4
1dc9 : 5f 85 5f a5 14 05 15 85  40
1dd1 : 15 a5 f7 a6 f8 85 f9 86  3c
1dd9 : fa a6 af f0 27 b1 f9 25  62
1de1 : 14 85 ba b1 f9 25 5e 0a  f3
1de9 : 05 ba 91 f9 08 38 a5 f9  bc
1df1 : e9 08 85 f9 b0 02 c6 fa  ab
1df9 : 28 ca f0 08 b1 f9 2a 91  7a
1e01 : f9 4c ed b4 b1 f9 aa 25  12
1e09 : 15 85 ba 8a 2a 25 5f 05  34
1e11 : ba 91 f9 a5 ae c5 02 f0  ca
1e19 : 06 20 02 c5 4c d2 b4 60  58
1e21 : 20 ac c4 38 a5 ae e5 02  3b
1e29 : f0 f5 48 20 bc ad a5 af  59
1e31 : d0 2f a5 b8 25 ba 85 bc  d1
1e39 : 49 ff 85 bd 68 aa a5 f7  fd
1e41 : 85 f9 a5 f8 85 fa 20 02  00
1e49 : c5 b1 f7 25 bc 85 b6 b1  c0
1e51 : f9 25 bd 05 b6 91 f9 ca  62
1e59 : d0 e4 b1 f7 25 bd 91 f7  7d
1e61 : 60 68 48 aa a5 f7 85 fb  85
1e69 : a5 f8 85 fc a5 f8 85 fa  b9
1e71 : a5 f7 85 f9 29 07 d0 0f  df

1e79 : 38 a5 f7 e9 39 85 f7 a5  aa
1e81 : f8 e9 01 85 f8 d0 02 c6  0a
1e89 : f7 b1 f7 25 b8 85 b6 b1  f1
1e91 : f9 25 b9 05 b6 91 f9 ca  a1
1e99 : d0 d2 b1 f7 25 b9 91 f7  94
1ea1 : 18 a5 fb 69 08 85 f7 a5  90
1ea9 : fc 69 00 85 f8 c6 af 30  ef
1eb1 : 15 f0 08 a9 ff 85 b8 84  8e
1eb9 : b9 d0 a6 a5 ba 85 b8 a5  3f
1ec1 : bb 85 b9 4c 62 b5 68 60  6d
1ec9 : 20 ac c4 20 69 c6 38 a5  6d
1ed1 : 02 e5 ae f0 38 48 a0 00  d8
1ed9 : 20 bc ad a5 af d0 2f a5  01
1ee1 : b8 25 ba 85 bc 49 ff 85  ac
1ee9 : bd 68 aa a5 f8 85 fa a5  2c
1ef1 : f7 85 f9 20 1f c5 b1 f7  04
1ef9 : 25 bc 85 b6 b1 f9 25 bd  af
1f01 : 05 b6 91 f9 ca d0 e4 b1  2f
1f09 : f7 25 bd 91 f7 60 68 48  e9
1f11 : aa a5 f7 85 fb a5 f8 85  18
1f19 : fc a5 f8 85 fa a5 f7 85  9e
1f21 : f9 29 07 c9 07 d0 0f a5  28
1f29 : f7 69 38 85 f7 a5 f8 69  f7
1f31 : 01 85 f8 4c 39 b6 e6 f7  91
1f39 : b1 f7 25 b8 85 b6 b1 f9  0f
1f41 : 25 b9 05 b6 91 f9 ca d0  11
1f49 : d0 b1 f7 25 b9 91 f7 18  cd
1f51 : a5 fb 69 08 85 f7 a5 fc  f8
1f59 : 69 00 85 f8 c6 af 30 15  18
1f61 : f0 08 a9 ff 85 b8 84 b9  63
1f69 : d0 a4 a5 ba 85 b8 a5 bb  78
1f71 : 85 b9 4c 0f b6 68 60 20  38
1f79 : 21 c6 20 69 c6 ad 00 db  c4
1f81 : 29 e0 85 bb 85 bc 20 46  3e
1f89 : c6 20 bf c5 48 98 48 86  7f
1f91 : bd 20 79 00 c9 2c d0 20  3e
1f99 : 20 73 00 20 3d c6 8a 29  fd
1fa1 : 07 aa bd af c7 85 bc 20  3e
1fa9 : 46 c6 20 3d c6 8a 29 07  76
1fb1 : aa bd af c7 85 bb a6 bd  6b
1fb9 : 68 85 5f 68 85 5e e4 ae  05
1fc1 : b0 03 4c 9b b7 38 a5 c1  d1
1fc9 : e5 ac aa a5 c2 e5 ad a8  c7
1fd1 : 18 8a 65 5e 85 5e 98 65  cc
1fd9 : 5f 85 5f 38 a5 02 e5 ae  39
1fe1 : 18 65 bd 85 bd 20 6b b8  c8
1fe9 : a5 ac a6 c1 85 c1 86 fb  3f
1ff1 : a5 ad a6 c2 85 c2 86 fc  f1
1ff9 : a5 ae a6 02 85 02 86 ba  d7
2001 : a5 5e a6 5f a4 bd 85 ac  12
2009 : 86 ad 84 ae a0 00 20 9a  1c
2011 : c6 85 bf 49 ff 85 af a5  e9
2019 : f7 85 f9 a5 f8 29 1f 05  65
2021 : bb 85 fa a5 fb a6 fc 85  06
2029 : ac 86 ad a5 ba 85 ae 20  0b
2031 : 9a c6 a5 f8 29 1f 05 bc  d0
2039 : 85 f8 b1 f7 25 be f0 09  c4
2041 : b1 f9 05 bf 91 f9 4c 50  e3
2049 : b7 b1 f9 25 af 91 f9 a5  b7
2051 : ad c5 c2 d0 14 a5 ac c5  58
2059 : c1 d0 0e a5 ba c5 02 d0  3e
2061 : 01 60 c6 bd c6 ba 4c 01  71
2069 : b7 a5 ac d0 02 c6 ad c6  d3
2071 : ac 06 be 90 0c 26 be a5  1a
2079 : f7 e9 07 85 f7 b0 02 c6  72
2081 : f8 06 bf 90 0c 26 bf a5  ba
2089 : f9 e9 07 85 f9 b0 02 c6  a4
2091 : fa a5 bf 49 ff 85 af 4c  fa
2099 : 3b b7 a5 ac a6 c1 85 c1  c1
20a1 : 86 fb a5 ad a6 c2 85 c2  60
20a9 : 86 fc a5 ae 85 ba 38 a5  47
20b1 : fb e5 c1 aa a5 fc e5 c2  c4
20b9 : a8 18 8a 65 5e 85 5e 98  79
20c1 : 65 5f 85 5f c9 01 d0 04  13
20c9 : a5 5e c9 40 90 03 4c 15  94
20d1 : c6 a0 00 a5 5e a6 5f 85  40
20d9 : ac 86 ad a5 bd 85 ae 20  eb
20e1 : 9a c6 85 bf 49 ff 85 af  42
20e9 : a5 f7 85 f9 a5 f8 29 1f  2f
20f1 : 05 bb 85 fa a5 fb a6 fc  63
20f9 : 85 ac 86 ad a5 ba 85 ae  cf
2101 : 20 9a c6 a5 f8 29 1f 05  34
2109 : bc 85 f8 b1 f7 25 be f0  82
2111 : 08 b1 f9 05 bf 91 f9 d0  23
2119 : 06 b1 f9 25 af 91 f9 a5  d6
2121 : ad c5 c2 d0 14 a5 ac c5  28
2129 : c1 d0 0e a5 ba c5 02 d0  0e
2131 : 01 60 e6 bd e6 ba 4c d4  f3
2139 : b7 a5 ac d0 02 c6 ad c6  a3
2141 : ac 06 be 90 0c 26 be a5  ea
2149 : f7 e9 07 85 f7 b0 02 c6  42
2151 : f8 06 bf 90 0c 26 bf a5  8a
2159 : f9 e9 07 85 f9 b0 02 c6  74
2161 : fa a5 bf 49 ff 85 af 4c  ca
2169 : 0c b8 a5 bd c9 c8 b0 0d  b2
2171 : a5 5f c9 01 d0 04 a5 5e  d9
2179 : c9 40 b0 01 60 4c 15 c6  f9
2181 : 20 b5 c5 20 46 c6 20 3d  87

2189 : c6 86 b6 20 46 c6 20 3d  da
2191 : c6 86 b7 20 46 c6 20 96  d4
2199 : c5 e0 01 d0 02 c0 68 90  12
21a1 : 04 a0 67 a2 01 98 e0 00  7c
21a9 : d0 28 c9 5a b0 0c a8 b9  cc
21b1 : 76 c9 85 b8 a9 00 85 b9  a8
21b9 : f0 3c e0 00 d0 14 c9 b4  3e
21c1 : b0 10 e9 59 a8 b9 76 c9  e5
21c9 : 49 ff 85 b8 a9 01 85 b9  b6
21d1 : d0 24 e0 01 d0 02 c9 0e  6c
21d9 : b0 0e e9 b3 a8 b9 76 c9  47
21e1 : 85 b8 a9 02 85 b9 d0 0e  f2
21e9 : e9 0e a8 b9 76 c9 49 ff  15
21f1 : 85 b8 a9 03 85 b9 a6 b8  cf
21f9 : bd 76 c8 a6 b6 20 01 a8  ba
2201 : 85 ba a5 b8 49 ff aa bd  1e
2209 : 76 c8 a6 b7 20 01 a8 85  3c
2211 : bb a5 b9 d0 1c 18 a5 ac  9a
2219 : 65 ba 85 c1 a5 ad 69 00  e2
2221 : 85 c2 38 a5 ae e5 bb 90  f4
2229 : 6a 85 02 20 95 b9 4c c0  b4
2231 : a1 c9 01 d0 20 18 a5 ac  c4
2239 : 65 ba 85 c1 a5 ad 69 00  02
2241 : 85 c2 18 a5 ae 65 bb b0  48
2249 : 4a c9 c8 b0 46 85 02 20  99
2251 : 95 b9 4c c0 a1 c9 02 d0  00
2259 : 1f a5 ac e5 ba 85 c1 a5  5d
2261 : ad e9 00 85 c2 18 a5 ae  94
2269 : 65 bb b0 27 c9 c8 b0 23  a9
2271 : 85 02 20 95 b9 4c c0 a1  f6
2279 : a5 ac e5 ba 85 c1 a5 ad  9d
2281 : e9 00 85 c2 38 a5 ae e5  5b
2289 : bb 90 08 85 02 20 95 b9  2a
2291 : 4c c0 a1 60 a5 c2 f0 fb  de
2299 : c9 02 b0 06 a5 c1 c9 40  60
22a1 : 90 f1 68 68 60 a0 00 ad  b7
22a9 : 00 db 84 f7 85 f8 a2 20  a2
22b1 : b1 f7 49 ff 91 f7 c8 d0  4e
22b9 : f7 e6 f8 ca d0 f2 60 20  21
22c1 : 3d c6 8a 29 07 aa bd af  45
22c9 : c7 85 bd 20 46 c6 20 7c  da
22d1 : c5 a9 35 85 01 ad 11 d0  cc
22d9 : 48 29 ef 8d 11 d0 20 ff  7b
22e1 : b9 a9 00 85 c1 85 ae a9  76
22e9 : 20 85 c2 a9 40 85 af 20  e1
22f1 : 58 c6 20 ff b9 a9 35 85  7d
22f9 : 01 68 8d 11 d0 60 a0 00  46
2301 : 84 f7 84 f9 a9 20 85 f8  85
2309 : a5 bd 85 fa a9 20 85 b6  6d
2311 : b1 f7 aa b1 f9 91 f7 8a  c0
2319 : 91 f9 c8 d0 f3 e6 f8 e6  1b
2321 : fa c6 b6 d0 eb 60 20 3d  03
2329 : c6 8a 29 07 aa bd af c7  46
2331 : 48 20 46 c6 20 7c c5 68  c2
2339 : a8 a2 00 8a 4c 61 c6 20  af
2341 : 3d c6 8a 29 07 aa bd af  c5
2349 : c7 85 f8 20 46 c6 20 3d  ab
2351 : c6 8a 29 07 aa bd af c7  6e
2359 : 85 fa 20 46 c6 20 3d c6  1c
2361 : 8a 29 03 85 b6 a0 00 84  6b
2369 : f7 84 f9 a9 20 85 b8 a6  b4
2371 : b6 f0 19 ca f0 11 ca f0  e4
2379 : 07 b1 f7 51 f9 4c 8f ba  37
2381 : b1 f7 31 f9 4c 8f ba b1  49
2389 : f7 11 f9 2c b1 f7 91 f9  22
2391 : c8 d0 dc e6 f8 e6 fa c6  15
2399 : b8 d0 d4 60 20 3d c6 8a  17
23a1 : 29 07 aa bd af c7 85 f8  f1
23a9 : 20 46 c6 20 3d c6 8a 29  28
23b1 : 07 aa bd af c7 85 fa 20  47
23b9 : 46 c6 20 3d c6 8a 29 03  7d
23c1 : 85 b6 a0 00 84 f7 84 f9  d7
23c9 : a9 20 85 b8 b1 f9 48 a6  54
23d1 : b6 f0 19 ca f0 11 ca f0  44
23d9 : 07 b1 f7 51 f9 4c ef ba  18
23e1 : b1 f7 31 f9 4c ef ba b1  ac
23e9 : f7 11 f9 2c b1 f7 91 f9  82
23f1 : 68 91 f7 c8 d0 d6 e6 f8  8a
23f9 : e6 fa c6 b8 d0 ce 60 20  6a
2401 : b5 c5 20 46 c6 20 31 c6  29
2409 : 84 b6 a0 00 84 b8 a4 b8  23
2411 : c4 b6 b0 5c e6 b8 a9 01  c5
2419 : 85 ba b1 22 e9 30 c9 04  fb
2421 : 90 02 c6 ba 29 03 85 b9  ef
2429 : a9 00 8d ad c7 ad ad c7  1b
2431 : cd ae c7 b0 d9 ee ad c7  b9
2439 : a4 ba d0 09 a6 ae e0 c8  85
2441 : b0 03 20 75 a1 a6 b9 bd  db
2449 : f1 c7 f0 13 30 09 e6 ac  fd
2451 : d0 0d e6 ad 4c 60 bb a5  19
2459 : ac d0 02 c6 ad c6 ac bd  06
2461 : f5 c7 f0 c9 30 05 e6 ae  d3
2469 : 4c 2e bb c6 ae 4c 2e bb  11
2471 : 60 20 3d c6 8e ae c7 20  c7
2479 : 46 c6 20 3d c6 8a 29 07  46
2481 : 0a 0a 0a a8 a2 00 b9 f9  2d
2489 : c7 9d f1 c7 c8 e8 e0 08  fb
2491 : 90 f4 60 20 3d c6 8a 29  3e
```

```
2499 : 07 aa 86 b8 8e aa c7 bd  87
24a1 : af c7 85 b6 20 46 c6 20  fc
24a9 : 3d c6 ad ab c7 f0 18 a9  e2
24b1 : 35 85 01 8a 29 07 aa bd  2b
24b9 : c7 c7 8d 00 dd bd bf c7  22
24c1 : 8d 18 d0 a9 30 85 01 38  67
24c9 : ad 00 db e5 b6 85 b6 a0  de
24d1 : 00 38 b9 00 db e5 b6 99  57
24d9 : 00 db c8 c0 c8 d0 f2 38  60
24e1 : ae 00 db 8a e9 80 8d 78  a1
24e9 : a8 8d b3 a9 ca a0 00 8a  41
24f1 : 99 93 a0 99 98 b0 e8 c8  f3
24f9 : c8 c8 c0 5e 90 f1 a6 b8  c6
2501 : bd b7 c7 aa 8e 16 a1 e8  d3
2509 : 8e 19 a1 e8 8e 1c a1 e8  cb
2511 : 8e 1f a1 60 20 3d c6 8a  bf
2519 : 29 07 aa bd af c7 85 f8  69
2521 : 85 b8 20 46 c6 20 3d c6  c3
2529 : 8a 29 07 aa bd af c7 85  e2
2531 : fa 85 b9 20 46 c6 20 3d  f6
2539 : c6 8a 09 01 85 b6 20 46  c2
2541 : c6 20 3d c6 8e ad c7 a0  f6
2549 : 00 84 f7 84 f9 84 fd 84  de
2551 : fb 84 fc a9 20 85 fe a0  6e
2559 : 00 b1 f7 aa b1 f9 91 f7  a6
2561 : 8a 91 f9 c8 cc ad c7 d0  46
2569 : fa 18 a5 f7 65 b6 85 f7  ea
2571 : 90 19 e6 f8 e6 fa e6 fc  42
2579 : a5 fc c9 20 90 0d a5 b8  8c
2581 : 85 f8 a5 b9 85 fa a9 00  fa
2589 : 85 fc 18 a5 f9 65 b6 85  f8
2591 : f9 18 a5 fb 65 b6 85 fb  99
2599 : a5 fd d0 02 c6 fe c6 fd  2d
25a1 : a5 fd 05 fe d0 b1 60 20  c2
25a9 : 79 00 aa d0 2e a9 35 85  f7
25b1 : 01 a9 31 a0 ea 8d 14 03  59
25b9 : 8c 15 03 a9 f0 8d 1a d0  4b
25c1 : ad a8 c7 f0 15 ad a7 c7  bf
25c9 : 8d 00 dd ad a6 c7 8d 11  84
25d1 : d0 ad a5 c7 8d 18 d0 ce  55
25d9 : a8 c7 60 c9 3a f0 ce 20  5d
25e1 : 3d c6 8e 3d c2 20 46 c6  a0
25e9 : 20 3d c6 8e 22 c2 8e 26  ea
25f1 : c2 8d 1a d0 20 46 c6 20  aa
25f9 : 3d c6 8a 30 16 29 07 aa  5e
2601 : bd bf c7 8d 2b c2 bd c7  91
2609 : c7 8d 35 c2 a9 3b 8d 30  47
2611 : c2 d0 12 ad a2 c7 8d 2b  6b
2619 : c2 ad a3 c7 8d 30 c2 ad  54
2621 : a4 c7 8d 35 c2 20 46 c6  87
2629 : 20 3d c6 8a 30 16 29 07  51
2631 : aa bd bf c7 8d 42 c2 bd  14
2639 : c7 c7 8d 4c c2 a9 3b 8d  52
2641 : 47 c2 d0 12 ad a2 c7 8d  8a
2649 : 42 c2 ad a3 c7 8d 47 c2  58
2651 : ad a4 c7 8d 4c c2 a9 0f  93
2659 : a0 c2 8d 14 03 8c 15 03  2f
2661 : a9 35 85 01 ad a8 c7 d0  07
2669 : 15 ee a8 c7 ad 11 d0 8d  da
2671 : a6 c7 ad 18 d0 8d a5 c7  09
2679 : ad 00 dd 8d a7 c7 ad 3c  37
2681 : c2 8d 12 d0 ad 11 d0 29  a1
2689 : 7f 8d 11 d0 a9 81 8d 1a  3e
2691 : d0 60 20 21 c6 20 69 c6  5e
2699 : a0 00 38 a5 02 e5 ae 4a  9b
26a1 : 85 bd a5 ae 48 a5 ac 48  39
26a9 : a5 ad 48 a5 c1 85 ac a5  32
26b1 : c2 85 ad a5 02 85 ae 20  9d
26b9 : 9a c6 86 b7 a5 f7 85 5e  3c
26c1 : a5 f8 85 5f 68 85 ad 68  6a
26c9 : 85 ac 20 9a c6 a5 f7 85  84
26d1 : f9 a5 f8 85 fa 68 85 ae  f2
26d9 : 20 9a c6 a5 5f c5 fa d0  5e
26e1 : 11 a5 f7 c5 5e d0 0b bd  8f
26e9 : e1 c7 a6 b7 2d e9 c7 4c  28
26f1 : 54 be 38 a5 5e e5 f9 85  6f
26f9 : b6 a5 5f e5 fa 4a a5 b6  1c
2701 : 6a 4a 4a 85 b6 18 bd e1  ba
2709 : c7 85 b9 49 ff 85 ba a6  8f
2711 : b6 a5 b9 d0 06 a9 ff 85  db
2719 : b9 84 ba 31 f7 85 bb b1  e7
2721 : f9 48 25 b9 85 bc b1 f7  b4
2729 : 25 ba 05 bc 91 f7 68 25  49
2731 : ba 05 bb 91 f9 a5 f7 69  0e
2739 : 08 85 f7 90 03 e6 f8 18  8f
2741 : a5 f9 69 08 85 f9 90 03  af
2749 : e6 fa 18 ca d0 c7 a6 b7  61
2751 : bd e9 c7 85 b9 49 ff 85  96
2759 : ba b1 f7 25 b9 85 bb b1  a8
2761 : f9 48 25 b9 85 bc b1 f7  f4
2769 : 25 ba 05 bc 91 f7 68 25  89
2771 : ba 05 bb 91 f9 e6 ae c6  ee
2779 : 02 a5 bd f0 05 c6 bd 4c  f1
2781 : a3 bd 60 20 21 c6 20 69  bb
2789 : c6 a0 00 38 a5 c1 e5 ac  00
2791 : 85 b6 a5 c2 e5 ad 4a 66  f5
2799 : b6 a5 ac 48 a5 ad 48 a5  8a
27a1 : c1 85 ac a5 c2 85 ad 20  54
```

```
27a9 : 9a c6 85 bf a5 f7 85 f9  24
27b1 : a5 f8 85 fa 68 85 ad 68  cd
27b9 : 85 ac 20 9a c6 a6 b6 b1  d0
27c1 : f7 25 be f0 17 b1 f9 25  4a
27c9 : bf d0 25 b1 f9 05 bf 91  5a
27d1 : f9 a5 be 49 ff 31 f7 91  02
27d9 : f7 4c f1 be b1 f9 25 bf  49
27e1 : f0 0e a5 bf 49 ff 31 f9  87
27e9 : 91 f9 b1 f7 05 be 91 f7  5f
27f1 : 46 be 90 0c 66 be a5 f7  1f
27f9 : 69 08 85 f7 90 02 e6 f8  6d
2801 : 06 bf 90 0c 26 bf a5 f9  77
2809 : e9 07 85 f9 b0 02 c6 fa  42
2811 : 8a f0 04 ca 4c c0 be a5  7f
2819 : ae c5 02 f0 05 e6 ae 4c  23
2821 : 9a be 60 c9 91 f0 07 c9  bc
2829 : e0 f0 03 4c 1b c6 e9 e0  1d
2831 : 8d a9 c7 4c 73 00 20 a3  0d
2839 : c5 20 b5 c5 20 ac c5 a9  06
2841 : 80 85 62 84 63 84 64 84  a2
2849 : 65 84 66 84 0d 20 9a c6  e4
2851 : 31 f7 f0 02 a0 81 84 61  e5
2859 : 60 00 00 00 00 00 00 00  ba
2861 : 00 00 00 00 00 00 00 00  62
2869 : 00 00 00 00 00 00 00 00  6a
2871 : 00 00 00 00 00 00 00 00  72
2879 : 00 00 00 00 00 00 00 00  7a
2881 : 00 00 00 00 00 00 00 00  82
2889 : 00 00 00 00 00 00 00 00  8a
2891 : 00 00 00 00 00 00 00 00  92
2899 : 00 00 00 00 00 00 00 00  9a
28a1 : 00 00 00 00 00 00 00 00  a2
28a9 : 00 00 00 00 00 00 00 00  aa
28b1 : 00 00 00 00 00 00 00 00  b2
28b9 : 00 00 00 00 00 00 00 00  ba
28c1 : 00 00 00 00 00 00 00 00  c2
28c9 : 00 00 00 00 00 00 00 00  ca
28d1 : 00 00 00 00 00 00 00 00  d2
28d9 : 00 00 00 00 00 00 00 00  da
28e1 : 00 00 00 00 00 00 00 00  e2
28e9 : 00 00 00 00 00 00 00 00  ea
28f1 : 00 00 00 00 00 00 00 00  f2
28f9 : 00 00 00 00 00 00 00 20  3a
2901 : 8e c6 20 01 a0 20 94 c6  05
2909 : a9 49 a0 c8 20 1e ab a9  8d
2911 : 99 a0 e4 20 2d e4 4c 86  70
2919 : e3 a6 7a a0 04 84 0f bd  1e
2921 : 00 02 10 07 c9 f1 f0 3e  e4
2929 : e8 d0 f4 c9 20 f0 37 85  61
2931 : 08 c9 22 f0 56 24 0f 70  68
2939 : 2d c9 3f d0 04 a9 99 d0  ca
2941 : 25 c9 30 90 04 c9 3c 90  0a
2949 : 1d 84 71 a0 00 84 0b 88  7a
2951 : 86 7a ca c8 e8 bd 00 02  60
2959 : 38 f9 9e a0 f0 f5 c9 80  30
2961 : d0 30 05 0b a4 71 e8 c8  f7
2969 : 99 fb 01 b9 fb 01 f0 57  b2
2971 : 38 e9 3a f0 04 c9 49 d0  a0
2979 : 02 85 0f 38 e9 55 d0 9f  d4
2981 : 85 08 bd 00 02 f0 df c5  2c
2989 : 08 f0 db c8 99 fb 01 e8  68
2991 : d0 f0 a6 7a e6 0b c8 b9  30
2999 : 9d a0 10 fa b9 9e a0 d0  9e
29a1 : b4 a0 ff ca c8 e8 38 bd  2f
29a9 : 00 02 f9 d1 c9 f0 f5 c9  f2
29b1 : 80 f0 af a6 7a e6 0b c8  07
29b9 : b9 d0 c9 10 fa b9 d1 c9  a7
29c1 : d0 e4 bd 00 02 10 9d 99  bd
29c9 : fd 01 c6 7b a9 ff 85 7a  0d
29d1 : 60 20 73 00 c9 f5 b0 08  3d
29d9 : c9 8b f0 25 c9 cc b0 06  1a
29e1 : 20 79 00 4c e7 a7 20 f0  66
29e9 : c0 20 94 c6 4c ae a7 e9  64
29f1 : cc 0a a8 b9 9f ca 48 b9  08
29f9 : 9e ca 48 20 8e c6 4c 73  4a
2a01 : 00 20 73 00 20 9e ad a5  e7
2a09 : 61 d0 06 20 3b a9 4c ae  e8
2a11 : a7 20 79 00 f0 f8 c9 a7  74
2a19 : d0 ba 20 73 00 b0 b5 20  59
2a21 : a0 a8 4c ae a7 a9 00 85  d1
2a29 : 0d 20 73 00 c9 ff f0 04  8c
2a31 : c9 f5 b0 06 20 79 00 4c  48
2a39 : 8d ae 20 44 c1 a2 37 86  c9
2a41 : 01 58 60 e9 f5 0a aa bd  99
2a49 : f1 ca 48 bd f0 ca 48 20  30
2a51 : 8e c6 4c 73 00 08 a2 00  8f
2a59 : 86 c7 28 10 0f c9 ff f0  f0
2a61 : 0b 24 0f 30 07 c9 cc b0  9c
2a69 : 06 4c 24 a7 4c f3 a6 e9  66
2a71 : cb aa ad a9 c7 85 c7 84  03
2a79 : 49 a0 ff ca f0 08 c8 b9  51
2a81 : d1 c9 10 fa 30 f5 c8 b9  e3
2a89 : d1 c9 30 05 20 47 ab d0  78
2a91 : f5 4c ef a6 78 48 8a 48  01
2a99 : 98 48 a5 01 48 a9 37 85  99
2aa1 : 01 a9 7f 8d 0d dd ac 0d  95
2aa9 : dd 30 3b 20 bc f6 20 e1  39
2ab1 : ff d0 33 20 d6 c1 20 a3  2c
```

```
2ab9 : fd 20 18 e5 68 a0 00 8c  2e
2ac1 : ab c7 84 d4 84 d8 b9 4a  96
2ac9 : c8 f0 07 20 d2 ff 78 c8  70
2ad1 : d0 f4 6c 02 a0 a9 95 a0  66
2ad9 : c1 8d 18 03 8d fa ff 8c  91
2ae1 : 19 03 8c fb ff 60 68 85  ce
2ae9 : 01 4c 72 fe ad ab c7 f0  c6
2af1 : 09 20 8e c6 20 52 a1 20  e2
2af9 : 94 c6 4c 8b e3 ad ab c7  5f
2b01 : f0 09 20 8e c6 20 52 a1  4a
2b09 : 20 94 c6 4c 83 a4 ad 19  f5
2b11 : d0 8d 19 d0 30 07 ad 0d  14
2b19 : dc 58 4c 31 ea ad 12 d0  61
2b21 : c9 00 b0 17 a9 00 8d 12  ee
2b29 : d0 a9 00 8d 18 d0 a9 00  2e
2b31 : 8d 11 d0 a9 00 8d 00 dd  d8
2b39 : 4c 81 ea a9 00 8d 12 d0  8c
2b41 : a9 00 8d 18 d0 a9 00 8d  c6
2b49 : 11 d0 a9 00 8d 00 dd 4c  16
2b51 : 81 ea 20 3d c6 a9 00 8d  cc
2b59 : 05 c3 8d e3 c3 85 5e 85  0c
2b61 : fe a9 01 85 fd 8a 10 06  a5
2b69 : ce 05 c3 ce e3 c3 29 07  93
2b71 : aa bd af c7 85 b6 a9 37  02
2b79 : 85 01 a9 00 20 bd ff a9  2c
2b81 : 04 a2 04 a0 ff 84 bd 20  47
2b89 : ba ff 20 c0 ff a9 00 85  bb
2b91 : ac 85 ad 85 ae 20 79 00  ee
2b99 : c9 2c d0 06 20 73 00 4c  a3
2ba1 : 52 c3 a9 28 85 bb a2 04  0d
2ba9 : 86 13 20 18 e1 a0 04 b9  6b
2bb1 : cf c7 20 d2 ff 88 10 f7  3b
2bb9 : a5 ae 85 b8 a0 00 84 b9  bd
2bc1 : 78 a9 30 85 01 18 a6 ae  93
2bc9 : a5 ac 29 f8 7d 00 da 85  7c
2bd1 : f7 a5 ad 7d 00 db 29 1f  78
2bd9 : 05 b6 85 f8 a0 00 b1 f7  7a
2be1 : a2 37 86 01 a4 b9 99 9a  94
2be9 : c7 e6 ae c8 84 b9 c0 07  0f
2bf1 : d0 ce a5 b8 85 ae a0 08  09
2bf9 : a9 00 a2 07 1e 9a c7 2a  56
2c01 : ca 10 f9 49 00 25 bd 09  ad
2c09 : 80 20 d6 c6 88 d0 e9 18  0f
2c11 : a5 ac 69 08 85 ac 90 03  6e
2c19 : e6 ad 18 c6 bb d0 99 a9  b1
2c21 : 00 85 ac 85 ad a5 ae 69  55
2c29 : 07 85 ae a9 0d 20 d6 c6  8e
2c31 : a5 bd c9 ff d0 10 a5 ae  a9
2c39 : c9 c4 b0 03 4c a3 c2 a9  31
2c41 : 0f 85 bd 4c a3 c2 20 d2  82
2c49 : ff 20 b5 ab a9 04 4c c3  af
2c51 : ff 20 3d c6 a9 37 85 01  f5
2c59 : a9 ff 85 14 8a d0 03 4c  ba
2c61 : a3 c2 a9 c7 85 ae a2 04  29
2c69 : 86 13 20 18 e1 a0 05 b9  2f
2c71 : d3 c7 20 d2 ff 88 d0 f7  02
2c79 : a5 14 49 ff 85 14 a5 ac  63
2c81 : a4 ad 85 f9 84 fa a5 f9  47
2c89 : a4 fa 85 ac 84 ad a5 14  16
2c91 : 85 15 a0 00 84 ba a9 07  9c
2c99 : 85 b9 78 a9 30 85 01 18  b1
2ca1 : a6 ae a5 ac 29 f8 7d 00  ee
2ca9 : da 85 f7 a5 ad 7d 00 db  77
2cb1 : 29 1f 05 b6 85 f8 a5 ac  92
2cb9 : 29 07 aa bd d9 c7 31 f7  59
2cc1 : f0 01 38 66 ba c6 b9 f0  b7
2cc9 : 10 a5 15 49 ff 85 15 d0  3c
2cd1 : ea e6 ac d0 ca e6 ad d0  b0
2cd9 : c6 a9 37 85 01 46 ba a5  6b
2ce1 : ba 49 00 25 bd 09 80 48  9b
2ce9 : 20 d6 c6 68 20 d6 c6 c6  94
2cf1 : ae a5 ae c9 ff d0 8f a9  6f
2cf9 : 0d 20 d6 c6 a5 ad d0 03  b6
2d01 : 4c 63 c3 a5 ac c9 3e b0  18
2d09 : 03 4c 63 c3 a5 bd 10 07  1a
2d11 : a9 0f 85 bd 4c 63 c3 4c  e2
2d19 : 47 c3 20 51 c4 20 46 c6  68
2d21 : 20 3d c6 a0 00 8a 0a 0a  36
2d29 : 0a 0a 85 f9 b1 f7 29 0f  76
2d31 : 85 fa 20 79 00 c9 2c d0  0b
2d39 : 0b 20 73 00 20 3d c6 8a  4d
2d41 : 29 0f 85 fa a5 f9 05 fa  e7
2d49 : a0 00 91 f7 60 4c 15 c6  97
2d51 : 20 bf c5 18 29 f8 7d 00  16
2d59 : da 85 f7 98 7d 00 db a8  9f
2d61 : 38 a5 f7 e9 00 85 f7 98  e4
2d69 : ed 00 db 4a 66 f7 4a 66  b3
2d71 : f7 4a 66 f7 ae aa c7 38  f6
2d79 : 7d b7 c7 85 f8 60 20 a3  cf
2d81 : c5 20 51 c4 a0 00 b1 f7  04
2d89 : 48 20 46 c6 20 3d c6 8a  68
2d91 : d0 06 68 4a 4a 4a 4a 24  30
2d99 : 68 29 0f a8 20 4f c6 4c  9f
2da1 : ac c5 20 94 c6 20 15 fd  88
2da9 : 4c 53 e4 20 21 c6 20 69  78
2db1 : c6 a5 ae a4 02 85 02 84  e7
```

Listing 1. (Fortsetzung)

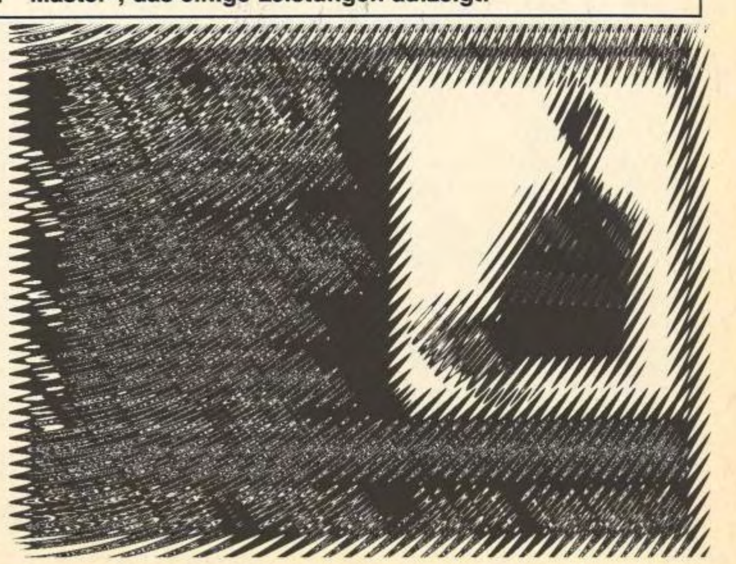
```
2db9 : ae a5 ad 4a a5 ac 6a 4a  ec
2dc1 : 4a 85 be a5 c2 4a a5 c1  cb
2dc9 : 6a 4a 4a 38 e5 be 85 af  bc
2dd1 : 85 bf 20 8e c6 a0 00 60  42
2dd9 : 48 8a 20 f7 c4 0a 0a 0a  46
2de1 : 0a 85 be 68 20 f7 c4 05  49
2de9 : be 60 20 3d c6 8a 29 07  fb
2df1 : 0a 0a 0a 0a 0a 60 38 e9  1d
2df9 : 3a b0 03 69 07 38 69 02  55
2e01 : 60 a5 f7 29 07 f0 05 c6  f0
2e09 : f7 c6 ae 60 38 a5 f7 e9  7f
2e11 : 39 85 f7 a5 f8 e9 01 85  ad
2e19 : f8 c6 ae 60 a5 f7 29 07  f9
2e21 : c9 07 b0 05 e6 f7 e6 ae  62
2e29 : 60 a5 f7 69 38 85 f7 a5  62
2e31 : f8 69 01 85 f8 e6 ae 60  11
2e39 : a5 ac d0 02 c6 ad c6 ac  f7
2e41 : 06 be b0 01 60 26 be a5  70
2e49 : f7 e9 07 85 f7 b0 f5 c6  12
2e51 : f8 4c 45 c5 e6 ac d0 02  94
2e59 : e6 ad 46 be b0 01 60 66  e1
2e61 : be a5 f7 69 08 85 f7 90  cb
2e69 : f5 e6 f8 d0 f1 48 20 94  34
2e71 : c6 68 20 d2 ff 48 20 94  ae
2e79 : c6 68 60 20 94 c6 20 d4  39
2e81 : e1 4c 8e c6 20 94 c6 20  07
2e89 : 9e ad 20 9b bc 20 8e c6  0e
2e91 : a5 65 a4 64 60 20 94 c6  85
2e99 : 20 9e ad 20 f7 b7 aa 4c  f8
2ea1 : 8e c6 20 94 c6 20 fa ae  e4
2ea9 : 4c 8e c6 20 94 c6 20 f7  e2
2eb1 : ae 4c 8e c6 20 bf c5 85  24
2eb9 : ac 84 ad 86 ae 60 a9 37  e6
2ec1 : 85 01 20 9e ad a5 66 30  a5
2ec9 : 4b a5 61 c9 91 b0 45 20  6c
2ed1 : 9b bc a5 65 a4 64 c0 01  53
2ed9 : d0 02 c9 40 b0 36 85 14  20
2ee1 : 84 15 20 fd ae 20 9e ad  79
2ee9 : a5 66 30 28 a5 61 c9 89  72
2ef1 : b0 22 20 9b bc 78 a9 30  c4
2ef9 : 85 01 a6 65 e0 c8 b0 14  94
2f01 : a5 14 a4 15 60 20 3d c6  06
2f09 : e0 03 b0 08 8a 6a 6a 6a  12
2f11 : 8d 98 c7 60 20 94 c6 4c  43
2f19 : 48 b2 20 94 c6 4c 08 af  a3
2f21 : 20 b5 c5 20 46 c6 20 bf  2c
2f29 : c5 85 c1 84 c2 86 02 60  db
2f31 : 20 94 c6 20 9e ad 20 82  2e
2f39 : b7 4c 8e c6 20 94 c6 20  95
2f41 : 9e b7 4c 8e c6 20 94 c6  ed
2f49 : 20 fd ae 4c 8e c6 20 94  66
2f51 : c6 20 a2 b3 4c 8e c6 20  db
2f59 : 94 c6 20 ed f5 4c 8e c6  a0
2f61 : 48 20 94 c6 68 4c d5 ff  f7
2f69 : a5 02 c5 ae b0 06 a4 ae  82
2f71 : 85 ae 84 02 a5 c2 a6 c1  3d
2f79 : c5 ad d0 02 e4 ac b0 0c  18
2f81 : a4 ad 85 ad 84 c2 a4 ac  5d
2f89 : 86 ac 84 c1 60 78 a9 30  8f
2f91 : 85 01 60 a9 37 85 01 58  38
2f99 : 60 a6 ae 18 a5 ac 29 f8  51
2fa1 : 7d 00 da 85 f7 a5 ad 7d  e4
2fa9 : 00 db 85 f8 a5 ac 29 07  8a
2fb1 : aa bd d9 c7 85 be 60 a6  c6
2fb9 : ae 18 a5 ac 29 f8 7d 00  c3
2fc1 : d8 85 f7 a5 ad 7d 00 d9  89
2fc9 : 85 f8 a5 ac 29 07 aa bd  ba
2fd1 : d9 c7 85 be 60 c9 0d d0  f1
2fd9 : 5b a5 5e c9 80 f0 13 a5  ff
2fe1 : fe d0 2e a5 fd c9 03 b0  23
2fe9 : 16 a5 5e 20 d2 ff c6 fd  b2
2ff1 : d0 f9 a9 00 85 5e 85 fd  86
2ff9 : 85 fe a9 0d 4c d2 ff a9  b8
3001 : 1a 20 d2 ff a5 fd 20 d2  50
3009 : ff a5 5e 20 d2 ff 4c f3  bd
3011 : c6 e6 fd a9 1a 20 d2 ff  ed
3019 : a5 fd 20 d2 ff a5 5e 20  06
3021 : d2 ff a9 1a 20 d2 ff a9  8c
3029 : ff 20 d2 ff a5 5e 20 d2  60
3031 : ff 4c f3 c6 c5 5e d0 07  cd
3039 : e6 fd d0 02 e6 fe 60 85  85
3041 : 5f a5 5e 48 a5 5f 85 5e  3c
3049 : a5 fe d0 25 a5 fd c9 03  be
3051 : b0 0d 68 20 d2 ff c6 fd  ea
3059 : d0 f9 a9 01 85 fd 60 a9  cd
3061 : 1a 20 d2 ff a5 fd 20 d2  b0
3069 : ff a9 01 85 fd 68 4c d2  28
3071 : ff e6 fd a9 1a 20 d2 ff  86
3079 : a5 fd 20 d2 ff 68 48 20  24
3081 : d2 ff a9 1a 20 d2 ff a9  ec
3089 : ff 20 d2 ff a9 01 85 fd  02
3091 : c6 fe 68 4c d2 ff 10 40  68
3099 : d0 ff ff ff ff ff ff ff  69
30a1 : ff 15 1b c7 00 00 00 00  eb
30a9 : 01 00 00 00 01 01 e0 20  86
30b1 : 40 60 80 e0 e0 e0 db 07  f0
30b9 : 5f 5b 7f db db db 78 28  f0
30c1 : 80 78 78 78 78 78 00 03  fc
30c9 : 02 02 00 00 00 00 50 00  0e
30d1 : 10 1b 08 28 00 10 1b 08  73
30d9 : 80 40 20 10 08 04 02 01  2e
30e1 : ff 7f 3f 1f 0f 07 03 01  8b
30e9 : 80 c0 e0 f0 f8 fc fe ff  93
30f1 : 00 01 00 ff ff 00 01 00  76
30f9 : 00 01 00 ff ff 00 01 00  7e
3101 : 01 01 ff ff ff 01 01 ff  8f
3109 : 01 00 ff 00 00 01 00 ff  12
3111 : 01 ff ff 01 01 01 ff ff  4a
3119 : 00 ff 00 01 01 00 ff 00  49
3121 : ff ff 01 01 01 ff ff 01  92
3129 : ff 00 01 00 00 ff 00 01  6b
3131 : ff 01 01 ff ff ff 01 01  f7
3139 : 30 31 32 33 34 35 36 37  29
3141 : 38 39 41 42 43 44 45 46  a6
3149 : 93 05 92 8e 11 1d 2a 2a  cc
3151 : 2a 20 43 4f 4d 4d [illegible] 44  4b
3159 : 4f 52 45 20 36 34 20 48  3d
3161 : 49 52 45 53 2d 4d 41 53  78
3169 : 54 45 52 20 56 31 2e 30  01
3171 : 20 2a 2a 2a 0d 00 00 01  49
3179 : 02 03 03 04 05 06 07 07  e9
3181 : 08 09 0a 0b 0b 0c 0d 0e  53
3189 : 0e 0f 10 11 12 12 13 14  6b
3191 : 15 15 16 17 18 19 19 1a  7c
3199 : 1b 1c 1c 1d 1e 1f 1f 20  05
31a1 : 21 22 22 23 23 24 25 26  51
31a9 : 27 28 29 29 2a 2b 2c 2c  59
31b1 : 2d 2e 2e 2f 30 31 31 32  1d
31b9 : 33 34 34 35 36 36 37 38  1c
31c1 : 39 39 3a 3b 3b 3c 3d 3d  92
31c9 : 3e 3f 40 40 41 42 42 43  75
31d1 : 44 44 45 46 46 47 48 48  a2
31d9 : 49 49 4a 4b 4b 4c 4d 4d  aa
31e1 : 4e 4f 4f 50 50 51 52 52  32
31e9 : 53 53 54 55 55 56 56 57  b6
31f1 : 57 58 59 59 5a 5a 5b 5b  92
31f9 : 5c 5c 5d 5e 5e 5f 5f 60  c6
3201 : 60 61 61 62 62 63 63 64  4e
3209 : 64 65 65 66 66 67 67 68  56
3211 : 68 68 69 69 6a 6a 6b 6b  b3
3219 : 6b 6c 6c 6d 6d 6e 6e 6e  64
3221 : 6f 6f 70 70 70 71 71 71  ad
3229 : 72 72 72 73 73 73 74 74  6d
3231 : 74 75 75 75 76 76 76 77  50
3239 : 77 77 77 78 78 78 78 79  79
3241 : 79 79 79 7a 7a 7a 7a 7b  81
3249 : 7b 7b 7b 7b 7c 7c 7c 7c  67
3251 : 7c 7d 7d 7d 7d 7d 7d 7d  50
3259 : 7e 7e 7e 7e 7e 7e 7e 7e  59
3261 : 7f 7f 7f 7f 7f 7f 7f 7f  61
3269 : 7f 7f 7f 7f 7f 7f 7f 7f  69
3271 : 7f 7f 7f 7f 7f 00 02 05  8a
3279 : 08 0b 0e 11 13 16 19 1c  2b
3281 : 1f 22 24 27 2a 2d 30 33  d3
3289 : 35 38 3b 3e 41 44 46 49  53
3291 : 4c 4f 52 55 57 5a 5d 60  43
3299 : 63 66 68 6b 6e 71 74 77  ea
32a1 : 79 7d 80 82 85 88 8a 8d  2b
32a9 : 90 93 96 99 9b 9e a1 a4  5a
32b1 : a7 aa ac af b2 b5 b8 bb  01
32b9 : bd c0 c3 c6 c9 cc ce d1  82
32c1 : d4 d7 da dd df e2 e5 e8  72
32c9 : eb ee f0 f3 f6 f9 fc ff  19
32d1 : 48 45 4c d0 49 4e 49 d4  bf
32d9 : 43 4c d3 43 4f 4c 4f d2  da
32e1 : 47 52 4f ce 47 52 4f 46  d0
32e9 : c6 4d 4f 44 c5 50 4c 4f  61
32f1 : d4 4c 49 4e c5 43 49 52  48
32f9 : 43 4c c5 42 4c 4f 43 cb  00
3301 : 42 4f d8 46 49 4c cc 53  bb
3309 : 45 54 4d 53 cb 43 48 41  b1
3311 : 4d 53 cb 53 54 4f 4d 53  01
3319 : cb 52 4f 4c cc 54 45 58  a0
3321 : d4 53 49 5a c5 45 43 4c  69
3329 : d3 4f 46 c6 41 52 c3 46  50
3331 : 43 49 52 43 4c c5 53 43  dd
3339 : 52 4f 4c cc 52 41 c4 52  c6
3341 : 45 56 45 52 d3 47 53 41  94
3349 : 56 c5 47 4c 4f 41 c4 43  76
3351 : 4f 4e 4e 45 43 44 53 57  da
3359 : 41 d0 46 49 47 55 52 c5  b1
3361 : 54 55 52 ce 50 41 47 c5  86
3369 : 44 55 50 4c 49 43 41 54  52
3371 : c5 45 46 46 45 4b d4 57  e4
3379 : 49 4e 44 4f d7 58 4d 49  ec
3381 : d2 59 4d 49 d2 43 4f 50  a1
3389 : d9 4f 50 54 49 4f ce 43  79
3391 : 53 45 d4 54 45 53 d4 43  0f
3399 : 54 45 53 d4 00 60 b0 00  c5
33a1 : a0 8c a0 f4 a0 23 a1 51  9a
33a9 : a1 05 c6 84 a1 bc a1 03  9b
33b1 : a5 12 a6 1c a7 13 a8 78  33
33b9 : aa b1 aa d8 aa f7 aa e8  e8
33c1 : ad 11 b0 86 b0 a2 c4 08  37
33c9 : b1 17 b3 fe b3 80 b8 a5  40
33d1 : b9 bf b9 26 ba 3f ba 9c  67
33d9 : ba ff ba 71 bb 93 bb 77  a6
33e1 : b6 14 bc a7 bc 92 bd 83  24
33e9 : be 52 c2 23 bf 1a c4 36  32
33f1 : bf 7e c4 00 07 0f 3f 3c  7f
33f9 : 78 70 70 70 70 78 3c 3f  0e
3401 : 0f 07 00 00 80 80 80 7e  9f
3409 : 7c 78 00 00 78 7c 7e 80  28
3411 : 80 80 00 aa aa aa aa 55  7c
```

Listing 1. »Hires-Master« (Schluß)

```
10 GRON:CLS:MODE1
20 FORI=0TO159STEP5:J=100+I/1.6
30 LINE0,J,I,199
40 LINE0,199-J,I,0
50 LINE319,J,319-I,199
60 LINE319,199-J,319-I,0
70 NEXT
80 SIZE2,3,16,0,0
90 TEXT64,80,"HIRES-MASTER"
100 TEXT98,120,"TASTE..."
110 LINE64,80+3*8,320-64,80+3*8
120 POKE198,0:WAIT198,1
130 ECLS20:MODE2:X=319:Y=199
140 FORK=0TO1:FORJ=1TO200
150 FORI=0TOYSTEPJ
160 LINEI,I,X-I,Y-I:LINEI,Y-I,X-I,I
170 NEXTI,J,K
180 FORI=1TO100:NEXT
190 MODE2:X=319:Y=199
200 FORI=0TOY
210 LINEI,I,X-I,Y
220 LINEX-I,Y-I,I,0
230 LINE0,Y-I,I,I
240 LINEX,I,X-I,Y-I
250 LINEI,I,X-I,Y-I
260 NEXT
270 MODE2:X=160:Y=100
280 FORK=0TO1:FORJ=1TO188:FORI=188TO0STEP-J
290 CIRCLEX,Y,I
300 NEXTI,J,K
310 MODE1:X=160:Y=100
320 FORI=100TO0STEP-4
330 CIRCLEX,Y,Y-I,I
340 NEXT
350 POKE198,0:WAIT198,1
360 ECLS15:MODE2:X=110:Y=50
370 FORI=0TO99STEP2
380 CIRCLEX+I/2,Y+I/2,I
390 NEXT
400 POKE198,0:WAIT198,1
410 ECLS15:MODE2:Y=100:A=63
420 FORJ=0TO1
430 FORI=0TO255
440 CIRCLEI,Y,IANDA
```

```
450 NEXT
460 NEXT
470 ECLS15:MODE2:Y=100:R=30*RND(1)+20
480 FORJ=0TO1
490 FORI=0TO319
500 CIRCLEI,Y,R
510 NEXT
520 NEXT
530 MODE2:POKE650,128:B=20
540 SIZE1,1,7,0,0
550 TEXT0,0,"IN DIESEM DEMO WERDEN 11610 KREISE"
560 TEXT0,10,"VERSCHIEDENER GROESSE IN NUR 180.9 S
EK."
570 TEXT0,20,"GEZEICHNET !!"
580 POKE198,0:WAIT198,1:POKE198,0
590 SIZE1,1,8,0
600 FORA=1TOB:CLS
610 FORX=BTO320-BSTEPB
620 FORY=BTO200-BSTEPB
630 GETA$:IFA$<>""THEN680
640 FORR=0TOBSTEPA
650 CIRCLEX,Y,R
660 NEXTR,Y,X,A
670 POKE198,0:WAIT198,1
680 ECLS15:MODE2
690 FORI=0TO319STEP2
700 LINEI,0,I,199:NEXT
710 FORI=0TO199STEP2
720 LINE0,I,319,I:NEXT
730 CONNECT0,4,0
740 FORI=1TO5:CONNECT4,0,0
750 FORJ=0TO188STEPI
760 CIRCLE160,100,J:NEXT:NEXT
770 POKE198,0:WAIT198,1
780 ECLS15:MODE2
790 S=10
800 FORX=0TO319-SSTEPS
810 BLOCKX,199,X+S-2,RND(1)*200:NEXT
820 POKE198,0:WAIT198,1
830 CLS
840 FORY=0TO199-SSTEPS
850 BLOCK0,Y,RND(1)*319,Y+S-2:NEXTY
860 POKE198,0:WAIT198,1
870 ECLS15:MODE2:GRON
880 FORK=0TO1:FORJ=1TO200:FORI=0TO199STEPJ
890 BOXI,I,319-I,199-I
900 NEXT:NEXT:NEXT
910 MODE1:P=π/180
920 FORI=0TO360STEP18
930 X=COS(I*P)*9
940 Y=SIN(I*P)*9
950 SIZE1,1,X+.5,Y+.5,I/90
960 TEXT160+X*3,100+Y*3,"HIRES-MASTER"
970 NEXT
980 POKE198,0:WAIT198,1:POKE198,0
990 MODE2
1000 FCIRCLE160,100,140
1010 ECLS15:MODE1:A$="HIRES-MASTER"
1020 FORI=1TO4:T=I*8
1030 SIZEI,I,T,0,I-1
1040 TEXT0,Y,A$
1050 TEXT0,199-Y-T,A$
1060 Y=Y+T
1070 NEXT
1080 SIZE4,4,32,0,0
1090 TEXT0,83,"**********"
1100 POKE198,0:WAIT198,1
1110 ECLS15:MODE1:A=360/99
1120 FORR=0TO99STEP3
1130 I=R*A
1140 ARC160,100,R,R,I,360-I
1150 NEXT
1160 POKE198,0:WAIT198,1
1170 POKE56,128:CLR:A=19:DIMA$(A)
1180 A$(0)="111122224444888811112222444488881111122
22444488881111222244448888"
1190 A$(1)="7777BBBBDDDDEEEE7777BBBBDDDDEEEE7777BB
BBDDDDEEEE7777BBBBDDDDEEEE"
1200 A$(2)="EEEEDDDDBBBB7777EEEEDDDDBBBB7777EEEEDD
DDBBBB7777EEEEDDDDBBBB7777"
1210 A$(3)="88884444222211118888444422221111888844
44222211118888444422221111"
1220 A$(4)="AAAA5555AAAA5555AAAA5555AAAA5555AAAA55
55AAAA5555AAAA5555AAAA5555"
1230 A$(5)="FFFF888888888888FFFF888888888888FFFF88
8888888888FFFF888888888888"
1240 A$(6)="FFFF808080808080808080808080808080FFFF80
80808080808080808080808080"
1250 A$(7)="0000000001800240024002401E78200420041E
78024002400240018000000000"
1260 A$(8)="47478383C5C5EEEE5C5C38387474EEEE474783
83C5C5EEEE5C5C38387474EEEE"
1270 A$(9)="888867670707070788887676707070708888867
6707070707888876767070707070"
1280 A$(10)="00007F7F7F7F7F7F0000F7F7F7F7F7F700007
F7F7F7F7F7F0000F7F7F7F7F7F7"
1290 A$(11)="88884444222255558888444422225555888884
44422225555888844442222555"
1300 A$(12)="38384444828292928282444438380101383844
44828292928282444438380101"
1310 A$(13)="FFFF8181BFBFA1A1A1A1A1A1A1A1FFFFFFFFF8
181BFBFA1A1A1A1A1A1A1A1FFFF"
1320 A$(14)="88885555222255558888555522225555888885
55522225555888855552222555"
1330 A$(15)="80804040202010100808040402020101010
2020404080810102020404080 80"
1340 A$(16)="1111222244448888888844442222111111112
22244448888888884444222211111"
1350 A$(17)="40042008101188224444228811100820044 00
820111022884444882210112008"
1360 A$(18)="22202220555088 8F55502220555088 8F55502
2205550888F5550222022202220"
1370 A$(19)="C827E44FF29FF93E7C7C3EF99FF24FE427C84
FE49FF23EF97C7CF93EF29FE44F"
1380 FORI=0TOA:R=INT(RND(1)*(A+1)):A$=A$(I):A$(I)=
A$(R):A$(R)=A$:T=FRE(0):NEXT
1390 ECLS15:MODE1
1400 FORI=0TOA:X=RND(1)*320:Y=RND(1)*200:R=RND(1)*
30+20
1410 MODE0:FCIRCLEX,Y,R:MODE1
1420 CIRCLEX,Y,R
1430 SETMSK0,A$(I):FILLX,Y
1440 POKE198,0:WAIT198,1:NEXT
1450 MODE1:ECLS15
1460 SETMSK0,"111111111111FFFF111111111111FFFF1111
11111111FFFF111111111111FFFF"
1470 FILL0,0:SIZE1,2,8,0,0
1480 FORI=0TO1
1490 FORJ=3TO0STEP-1
1500 H=INT(RND(1)*30)
1510 FORK=0TOH
1520 DUPLICATE7+J*80+K,43+I*72-K,38+J*80+K,70+I*72
-K,8+J*80+K,42+I*72-K
1530 NEXT
1540 REM HEX$0:RESUME8+J*80+K,42+I*72-K,39+J*80+K,
69+I*72-K:HEX$1
1550 REMTRAP8+J*80+K,42+I*72-K,39+J*80+K,69+I*72-K
1560 TEXT8+J*80+K+(3-LEN(STR$(H)))*4,48+I*72-K,STR
$(H)
1570 NEXT:NEXT
1580 POKE198,0:WAIT198,1
1590 ECLS15:MODE2:C=1.6:X=319:YY=199
1600 FORI=0TO160STEP2:Y=INT(I/C)
1610 BOXI,Y,X-I,YY-Y
1620 IFI>100THEN:ROLL0,110,60,210,140
1630 NEXT
1640 SIZE2,2,14,0
1650 POKE198,0:WAIT198,1
1660 ECLS15:PAGE0,0:COLOR1,0:CLS:PAGE3,3:CLS:COLOR
1,0:MODE2
1670 FORI=0TO187:PAGE(IAND1)*3,((I+1)AND1)*3:FCIRC
LE160,100,I:NEXT
1680 POKE198,0:WAIT198,1:PAGE0,0
```


Listing 2. Demo-Programm zur Grafikerweiterung »Hires-Master«, das einige Leistungen aufzeigt.



```
4 REM *** BILDSCHIRM/VARIABLEN INIT ****  <103>
5 POKE 53281,0:POKE 53280,0:POKE 646,1    <202>
10 PRINT CHR$(142)CHR$(8)"{CLR,RVSON,13SPA
   CE}HIRES - MASTER{13SPACE}";           <157>
15 T$="0123456789ABCDEF"                  <048>
20 PRINT"{RVSON,10SPACE}FILL - MUSTER EDIT
   OR{10SPACE}"                           <004>
30 FOR I=1 TO 16:PRINT"{YELLOW,5SPACE}....
   ............":NEXT:X=1:Y=1:POKE 1149,43
   :MA=46:TX=1                            <193>
35 PRINT"{PURPLE,8SPACE}MG{2SPACE}MG{2SPAC
   E}MG{3SPACE}":TY=1                     <059>
40 PRINT"{HOME,3DOWN}";TAB(23);"{RVSON,WHI
   TE,4SPACE}BEFEHLE:{4SPACE,RVOFF}"      <083>
41 PRINT TAB(23);"{RED}STEUERUNG:CURSOR"  <082>
42 PRINT TAB(23);"{10SPACE}TASTEN"        <023>
43 PRINT TAB(23);"PUNKT SETZEN:' '"       <075>
45 PRINT TAB(23);"MEHR BEFEHLE:{RVSON,SPAC
   E}B{SPACE,RVOFF,DOWN}"                 <087>
46 PRINT TAB(23);"{RVSON}E{RVOFF,SPACE}END
   E{10SPACE}"                            <002>
47 PRINT TAB(23);"{RVSON},{RVOFF,SPACE}ZEI
   LENANFANG{2SPACE}"                     <173>
48 PRINT TAB(23);"{RVSON}.{RVOFF,SPACE}ZEI
   LENENDE{4SPACE}"                       <086>
49 PRINT TAB(23);"{RVSON}D{RVOFF,SPACE}DAT
   ENAUSGABE{2SPACE}"                     <225>
50 PRINT TAB(23);"{RVSON}X{RVOFF,SPACE}X-T
   AB SETZEN{2SPACE}"                     <207>
51 PRINT TAB(23);"{RVSON}Y{RVOFF,SPACE}Y-T
   AB SETZEN{2SPACE}"                     <228>
52 PRINT TAB(23);"{RVSON}A{RVOFF,SPACE}X-T
   AB SPRINGEN"                           <002>
53 PRINT TAB(23);"{RVSON}B{RVOFF,SPACE}Y-T
   AB SPRINGEN"                           <023>
54 PRINT TAB(23);"{RVSON}Z{RVOFF,SPACE}&{S
   PACE,RVSON}S{RVOFF,SPACE}:{7SPACE}"    <171>
55 PRINT TAB(23);"NORMAL :LOESCHEN"       <083>
56 PRINT TAB(23);"<SHIFT>:SETZEN{2SPACE}" <108>
57 PRINT TAB(23);"<C=>{3SPACE}:INVERT. "  <080>
98 :                                      <074>
99 REM **** STEUER-TASTATUR-ABFRAGE ****  <110>
100 A$="":POKE 198,0:WAIT 198,1:GET A$    <069>
104 IF A$<>"{RIGHT}"THEN 130              <150>
105 IF X=16 AND Y=16 THEN GOSUB 1000:X=1:Y
    =1:GOTO 500                           <066>
110 IF X=16 THEN GOSUB 1000:X=1:Y=Y+1:GOTO
     500                                  <030>
120 IF X<16 THEN GOSUB 1000:X=X+1:GOTO 500 <182>
130 IF A$="B"THEN POKE 53280,2:GOTO 1900  <052>
135 IF A$<>"{LEFT}"THEN 165               <005>
150 IF X=1 AND Y>1 THEN GOSUB 1000:X=16:Y=
    Y-1:GOTO 500                          <005>
160 IF X>1 THEN GOSUB 1000:X=X-1:GOTO 500 <126>
165 IF A$<>"{DOWN}"THEN 185               <183>
170 IF Y=16 THEN GOSUB 1000:Y=1:GOTO 500  <249>
180 IF Y<16 THEN GOSUB 1000:Y=Y+1:GOTO 500 <059>
185 IF A$<>"{UP}"THEN 202                 <115>
190 IF Y=1 THEN GOSUB 1000:Y=16:GOTO 500  <028>
200 IF Y>1 THEN GOSUB 1000:Y=Y-1:GOTO 500 <078>
202 IF A$<>" "THEN 499                    <212>
203 IF X=16 AND MA=81 AND Y=16 THEN GOSUB
    1000:X=1:Y=1:MA=46:GOTO 500           <210>
204 IF X=16 AND MA=46 AND Y=16 THEN GOSUB
    1000:X=1:Y=1:MA=81:GOTO 500           <163>
205 IF X=16 AND MA=46 THEN GOSUB 1000:X=1:
    Y=Y+1:MA=81:GOTO 500                  <065>
210 IF X=16 AND MA=81 THEN GOSUB 1000:X=1:
    Y=Y+1:MA=46:GOTO 500                  <034>
215 IF MA=46 THEN GOSUB 1000:X=X+1:MA=81:G
    OTO 500                               <136>
220 IF MA=81 THEN GOSUB 1000:X=X+1:MA=46:G
    OTO 500                               <017>
499 A$="":GOTO 100                        <091>
500 MN=PEEK(1108+X+Y*40):POKE 1108+X+Y*40,
    43:POKE 1108+XA+YA*40,MA:A$="":MA=MN  <169>
510 GOTO 100                              <184>
998 :                                     <212>
999 REM *** UPROG X -> XA  Y -> YA *****  <233>
1000 XA=X:YA=Y:RETURN                     <141>
1897 :                                    <095>
1898 REM ****  ZUSAETZLICHE BEFEHLE ****  <087>
1900 PRINT"{HOME,23DOWN,6SPACE,RVSON}BEFEH
     LSEINGABE{RVOFF}"                    <203>
1950 POKE 198,0:WAIT 198,1:GET A$         <096>
2000 IF A$<>"{CLR}"THEN 2010              <134>
2005 X=1:Y=1:GOSUB 1000:PRINT"{HOME,2DOWN}
     ":FOR I=1 TO 16:PRINT"{YELLOW,5RIGHT}
     ................":NEXT               <084>
2006 MA=43:GOTO 2600                      <006>
2010 IF A$="{HOME}"THEN GOSUB 1000:X=1:Y=1
     :GOTO 2600                           <026>
2020 IF A$="E"THEN INPUT"{HOME,22DOWN}WIRK
     LICH BEENDEN";F$:IF F$="J"THEN END   <080>
2025 PRINT"{HOME,22DOWN,38SPACE}":F$=""   <068>
2030 IF A$=","AND X<>1 THEN GOSUB 1000:X=1
     :GOTO 2600                           <108>
2040 IF A$="."AND X<>16 THEN GOSUB 1000:X=
     16:GOTO 2600                         <092>
2050 IF A$="Z"THEN FOR J=0 TO 15:POKE 1109
     +Y*40+J,46:NEXT:J=0:MA=46:GOTO 2700  <023>
2051 IF A$="Ẕ"THEN FOR J=0 TO 15:POKE 1109
     +Y*40+J,81:NEXT:J=0:MA=81:GOTO 2700  <037>
2052 IF A$<>"Z̄"THEN 2059                  <152>
2053 FOR J=0 TO 15:IF PEEK(1109+Y*40+J)=46
      THEN POKE 1109+Y*40+J,81:GOTO 2055  <016>
2054 IF PEEK(1109+Y*40+J)=81 THEN POKE 110
     9+Y*40+J,46                          <007>
2055 NEXT:J=0:IF MA=46 THEN MA=81:GOTO 270
     0                                    <006>
2056 IF MA=81 THEN MA=46:GOTO 2700        <215>
2058 IF A$="S"THEN FOR J=1 TO 16:POKE 1109
     +J*40+X-1,46:NEXT:J=0:MA=46:GOTO 2700 <144>
2059 IF A$="S̲"THEN FOR J=1 TO 16:POKE 1109
     +J*40+X-1,81:NEXT:J=0:MA=81:GOTO 2700 <005>
2060 IF A$="X"THEN TX=X:PRINT"{HOME,2DOWN,
     24SPACE}":POKE 1108+X,93:GOTO 2700   <060>
2061 IF A$<>"S̄"THEN 2070                  <160>
2062 FOR J=1 TO 16:IF PEEK(1109+J*40+X-1)=
     46 THEN POKE 1109+J*40+X-1,81:GOTO 20
     65                                   <052>
2063 IF PEEK(1109+J*40+X-1)=81 THEN POKE 1
     109+J*40+X-1,46                      <164>
2065 NEXT:J=0:IF MA=46 THEN MA=81:GOTO 270
     0                                    <016>
2066 IF MA=81 THEN MA=46:GOTO 2700        <225>
2070 IF A$="Y"THEN TY=Y:FOR J=1 TO 17:POKE
      1107+J*40,96:NEXT:J=0:POKE 1107+Y*40
     ,64:GOTO 2700                        <150>
2080 IF A$="A"AND X<>TX THEN GOSUB 1000:X=
     TX:GOTO 2600                         <028>
2090 IF A$="B"AND Y<>TY THEN GOSUB 1000:Y=
     TY:GOTO 2600                         <116>
2100 IF A$=CHR$(13)THEN GOSUB 1000:X=1:Y=Y
     +1:GOTO 2600                         <076>
2110 IF A$<>"I"THEN 2400                  <204>
2114 :                                    <058>
2115 REM *** BEFEHL ' INVERTIEREN ' ****  <087>
2120 POKE 1108+X+Y*40,MA                  <109>
2130 FOR I=1 TO 16                        <250>
2140 FOR J=1 TO 16                        <012>
2150 IF PEEK(1108+I+J*40)=46 THEN POKE 110
     8+I+J*40,81:GOTO 2170                <214>
2160 IF PEEK(1108+I+J*40)=81 THEN POKE 110
     8+I+J*40,46                          <081>
2170 NEXT                                 <148>
2180 NEXT                                 <158>
2190 IF MA=46 THEN MA=81:GOTO 2210        <121>
2200 IF MA=81 THEN MA=46                  <052>
2210 A$="{RIGHT}":GOTO 105                <132>
2398 :                                    <088>
2399 REM *** DATENBERECHNUNG        ****  <126>
2400 IF A$<>"D"THEN 2500                  <251>
2404 PRINT"{HOME,23DOWN,6SPACE,RVSON}DATEN
     BERECHNUNG{RVOFF,3UP}"               <019>
2405 POKE 1108+X+Y*40,MA:W=0              <245>
2410 FOR Z=1 TO 16                        <156>
2412 :FOR C=0 TO 3                        <141>
2414 : FOR S=1 TO 4                       <209>
2416 :  IF PEEK(1108+Z*40+C*4+S)=81 THEN W
     =W+2↑(4-S)                           <020>
2418 :                                    <108>
2420 : NEXT:TE$=MID$(T$,W+1,1):S$=S$+TE$:P
     RINT TE$;:W=0:TE$=""                 <081>
2422 :NEXT                                <132>
2424 NEXT:S$=""                           <253>
2425 POKE 1108+X+Y*40,43                  <006>
2500 PRINT"{HOME,23DOWN,24SPACE}":POKE 532
     80,0:GOTO 100                        <178>
2600 PRINT"{HOME,23DOWN,24SPACE}":POKE 532
     80,0:GOTO 500                        <028>
2700 PRINT"{HOME,23DOWN,24SPACE}":POKE 532
     80,0:A$="{RIGHT}":GOTO 110           <244>
9999 PRINT"{CLR}"CHR$(9):POKE 53280,0:POKE
      646,1:LIST 2100-2299                <088>
```

Listing 3. Der Fill-Mustereditor für Hires-Master

Simplex optimiert lineare Zusammenhänge

Mit Simplex 64 können Sie eine Vielzahl von Problemen aus der Betriebswirtschaft, der Technik und dem häuslichen Bereich ohne großen Zeitaufwand sicher lösen. Das Ergebnis stellt die bestmögliche Problemlösung dar.

Ohne Einschränkungen ist das Programm Simplex 64 (Listing 1) auf einem C 64 in Verbindung mit einem Drucker MPS 801 lauffähig. Ein 80-Zeichen-Bildschirm (Hardware oder Software) erleichtert die Bearbeitung umfangreicher Aufgaben.

Eine grundlegende Voraussetzung ist bei der Anwendung der Simplex-Methode zu beachten; die dem Problem entsprechenden funktionalen Zusammenhänge müssen durch lineare Gleichungen oder Ungleichungen zu beschreiben sein.

Anhand eines einfachen Beispiels aus der Arbeitsvorbereitung eines Produktionsbetriebs sollen im folgenden die Übersetzung eines Problems in ein mathematisches Modell, sowie das Prinzip der Simplex-Methode erläutert werden.

Beispiel 1

Der Betrieb stellt zwei unterschiedliche Produkte P1 und P2 her. Aus der Differenz der Erlöse und der direkten Kosten (Rohmaterial) ergeben sich die folgenden Deckungsbeiträge:

300 Mark/Stück für P1
600 Mark/Stück für P2

Für die Produktion stehen drei verschiedene Maschinen A, B und C zur Verfügung. Die Herstellung einer Einheit von P1 erfordert eine zweistündige Bearbeitung auf Maschine A und eine dreistündige auf Maschine B. Für P2 müssen 6 Maschinenstunden angesetzt werden - drei auf Maschine A und drei auf Maschine C. Aufgrund unterschiedlich langer Wartungszeiten ergeben sich verschiedene Nutzungskapazitäten für die Maschinen.

190 Stunden für Maschine A
170 Stunden für Maschine B
180 Stunden für Maschine C

Die fixen Kosten des Unternehmens im untersuchten Zeitraum belaufen sich auf 2700 Mark. Der Gewinn aus dem Verkauf der beiden Produkte soll optimiert, das heißt maximiert werden.

Das mathematische Modell

Die angenommene Problemstellung läßt sich durch ein System von linearen Gleichungen und Ungleichungen beschreiben - dem mathematischen Modell. Die Zielfunktion ergibt sich aus der Forderung nach maximalem Gewinn (Maximumaufgabe).

G(max) = 300*X1 + 600*X2 - 2700

Die sogenannten Strukturvariablen X1 und X2 bedeuten darin die, im betrachteten Zeitraum hergestellten Mengen der Produkte P1 und P2.

Wären die produzierbaren Mengen unbegrenzt, so würde auch der Gewinn ins Unendliche wachsen. Einschränkungen (Restriktionen) der Produktion ergeben sich aus den zur Verfügung stehenden Nutzungskapazitäten der Maschinen A, B und C in Relation zu den notwendigen Bearbeitungszeiten für die zwei Produkte. Im Modell wird dieser Zusammenhang durch drei Ungleichungen (3 Maschinen) beschrieben.

2*X1 + 3*X2 ≤ 190
3*X1 + 0*X2 ≤ 170
0*X1 + 3*X2 ≤ 180

Ferner ist zu beachten, daß keine negativen Mengen produziert werden können; es ergeben sich die Nichtnegativitätsbedingungen für X1 und X2

X1 ≥ 0
X2 ≥ 0

Grafische Lösung

Bei zweidimensionalen Problemen, also Problemen mit nur zwei Strukturvariablen, besteht die Möglichkeit der grafischen Lösungsfindung. Zum besseren Verständnis der Verfahrensweise bei einer Optimierung soll die zeichnerische Methode anhand des Beispiels 1 kurz vorgestellt werden (Bild 1).

Auf den Achsen eines karthesischen Koordinatensystems werden X1 und X2 aufgetragen. Die Restriktionsgleichungen oder Ungleichungen können als Geraden in das Koordinatensystem eingezeichnet werden. Handelt es sich um Ungleichungen, so trennen sie einen zulässigen von einem unzulässigen Lösungsbereich. Der Beginn des unzulässigen Bereichs wird durch Schraffur entlang der Gerade gekennzeichnet. Entsprechend werden auch die Nichtnegativitätsbedingungen behandelt - Schraffur der Abszisse und Ordinate (Koordinatenachsen). Somit erhält man einen geschlossenen zulässigen Lösungsbereich, innerhalb dessen jeder Punkt als Lösung in Frage kommt. Gesucht ist jedoch der optimale Punkt, also das Produktionsprogramm mit dem maximalen Gewinn. Um diesen Punkt aufzufinden, zeichnet man die Zielfunktion ein. Da die Zielfunktion außer den Strukturvariablen X1 und X2

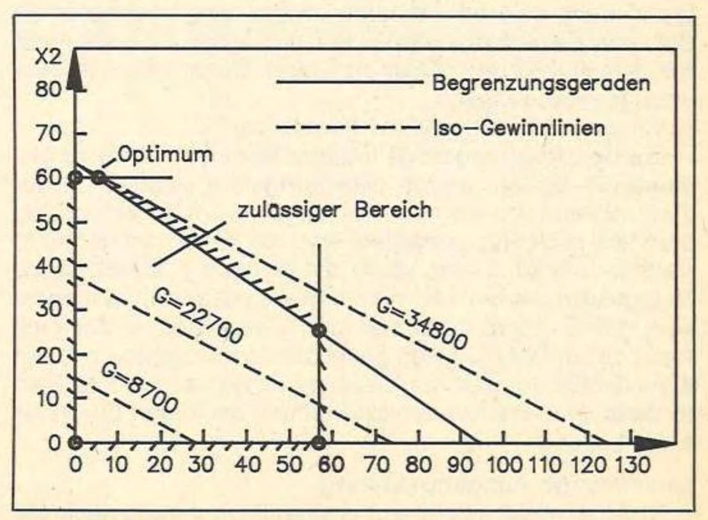


Bild 1. Grafische Lösung zum ersten Beispiel

noch den Gewinn G als Variable enthält, kann sie im zweidimensionalen Koordinatensystem nur dargestellt werden, wenn man für G feste Werte annimmt. Für verschiedene Werte von G ergeben sich unterschiedliche, einander parallele Geraden, die sogenannten »Iso-Gewinnlinien« (Linien gleichen Gewinns). Vorstellbar ist auch, daß eine Gewinnlinie ausgehend vom Koordinatenpunkt stetig parallel verschoben wird. Das gesuchte Gewinnmaximum liegt an der Stelle, an der, der durch den zulässigen Lösungsbereich gehende Teil der Gewinnlinie auf einen Punkt zusammenschrumpft. Man erkennt, daß der optimale Punkt in jedem Fall ein Eckpunkt des Lösungsbereichs sein muß. Nur wenn eine Begrenzungsgerade parallel zu der Gewinnlinie verläuft, kann das Gewinnmaximum aus einer Geraden (Punktmenge) bestehen; man bezeichnet dies als Entartungsfall. Auch im Entartungsfall gehören aber die entsprechenden Eckpunkte zur Optimallösung, so daß die zuvor gewonnene Erkenntnis selbst in diesem Fall ihre Gültigkeit behält.

Berechnung nach der Simplex-Methode

Das Prinzip der rechnerischen Lösungsfindung nach der Simplex-Methode besteht darin, ausgehend vom Ausgangspunkt (normalerweise Koordinatenursprung), iterativ (schrittweise) von Eckpunkt zu Eckpunkt zu springen bis das Optimum erreicht ist.

Da das Rechnen mit Ungleichungen umständlich ist, soll zunächst das mathematische Modell zum Beispiel 1 geringfügig modifiziert werden.

Durch das Hinzufügung sogenannter »Schlupfvariablen« wird das Ungleichungssystem der Restriktionen in ein Gleichungssystem überführt.

Y1 + 2*X1 + 3*X2 = 190
Y2 + 3*X1 + 0*X2 = 170
Y3 + 0*X1 + 3*X2 = 180

Die Schlupfvariablen Y1, Y2 und Y3 stellen die ungenutzten Kapazitäten der Maschinen A, B und C dar. Die Nichtnegativitätsbedingungen müssen nun auf die Schlupfvariablen ausgedehnt werden.

X1,X2,Y1,Y2,Y3 ≥ 0

Das gewonnene Gleichungssystem ohne die Zielfunktion besteht aus drei Gleichungen mit fünf Unbekannten, ist also unbestimmt. Somit scheidet eine vollständige Lösung, zum Beispiel mit dem Gaußschen Algorithmus, aus.

Es gibt unendlich viele Lösungen, von denen, wie wir wissen, nur die Eckpunkte zur Bestimmung des Optimums relevant sind. Ein Eckpunkt (beim zweidimensionalen Problem) stellt den Schnittpunkt zweier Begrenzungslinien dar; das Erreichen eines Eckpunkts ist gleichbedeutend mit dem Nullsetzen zweier Variablen. Damit wird das Gleichungssystem lösbar!

Noch einmal in allgemeiner Formulierung:

Zur Beschreibung eines n-dimensionalen Problems existiere ein System aus m Gleichungen einschließlich der Zielfunktion. Von den »m+n« Variablen (n Strukturvariablen und m Schlupfvariablen) sind die Werte von genau m Variablen bestimmbar, wenn die übrigen n Variablen zu Null gesetzt werden. An dieser Stelle sei darauf hingewiesen, daß G (Gewinn) formal die Schlupfvariable der Zielfunktion ist. Die n zu Null gesetzten Unbekannten werden als »Nichtbasisvariablen« bezeichnet, während die übrigen m Variablen »Basisvariablen« heißen; sie bilden die Basis einer Lösung.

Unzulässige Ausgangslösung

Üblicherweise geht man bei einer Berechnung nach der Simplex-Methode vom Koordinatenursprung (Nullpunkt) als Eckpunkt für die Ausgangslösung aus. Es besteht die Möglichkeit, daß der Nullpunkt außerhalb des zulässigen Lösungsbereichs der Problemstellung liegt. Dieser Fall ist im Programm SIMPLEX 64 entsprechend berücksichtigt; die Lösungsfindung wird nicht beeinträchtigt.

Gleichungen als Restriktionen

Bei bestimmten Problemstellungen kann es vorkommen, daß neben Ungleichungen auch Gleichungen als Restriktionen auftreten. Es erscheint zunächst sinnlos einer Gleichung eine Schlupfvariable zuzuordnen. Die Schlupfvariable hat aber auch die Aufgabe, in der Ausgangslösung, in der alle Strukturvariablen zu Null gesetzt werden, den Wert der rechten Seite (RS) der betreffenden (Un-)Gleichung anzunehmen, so daß diese erfüllt ist. Eine Gleichung, die nur Strukturvariablen (also keine Schlupfvariablen) enthält, ist in der Ausgangslösung nicht erfüllt, es sei denn ihre rechte Seite (RS) wäre Null. Daher ist im Programm SIMPLEX 64 vorgesehen, daß sowohl jeder Ungleichung, als auch jeder Gleichung eine Schlupfvariable zugeordnet wird. Eine Lösung ist aber nur dann zulässig, wenn die Schlupfvariablen von Gleichungen zu Null, das heißt aus der Basis entfernt werden. Zu diesem Zweck sind diese Variablen bei der Eingabe zu sperren.

Freie Strukturvariablen

Bei einigen Optimierungsaufgaben treten Strukturvariablen auf, für die die Nichtnegativitätsbedingungen keine Gültigkeit besitzen. Sie werden als freie Strukturvariablen bezeichnet und müssen bei der Berechnung endgültig in die Basis gelangen. Zu diesem Zweck ist es erforderlich, derartige Variablen bei der Eingabe zu kennzeichnen.

Minimumprobleme

Im Beispiel 1 wurde eine typische Maximumaufgabe beschrieben. Der Zielfunktionswert G (Gewinn) sollte maximiert werden. Häufig sind auch jene Fälle, in denen das Optimum ein Minimum darstellt; zum Beispiel sollen die Kosten minimiert werden. Da das Programm für die Bearbeitung von Maximumproblemen ausgelegt ist, empfiehlt es sich, bei Minimumaufgaben das mathematische Modell entsprechend anzupassen. Sollen die Kosten minimiert werden, so maximiert man statt dessen die negativen Kosten.

Beispiel 2:

	Minimumaufgabe
Zielfunktion	Z(min) - 3*X1 - 2*X2 = 0
Restriktionen	1*X1 + 1*X2 ≥ 30
	2*X1 + 1*X2 ≥ 50
	1*X1 + 3*X2 ≥ 40
Nichtnegative Bedingungen	X1,X2 ≥ 0
	Analoge Maximumaufgabe
Zielfunktion	Z(max) + 3*X1 + 2*X2 = 0
	Z(max) = -Z(min)
Restriktionen	-1*X1 - 1*X2 ≤ -30
	-2*X1 - 1*X2 ≤ -50
	-1*X1 - 3*X2 ≤ -40
Nichtnegative Bedingungen	X1,X2 ≤ 0

Simplex-Tableau (Standardein- und -ausgabe)

Um eine Standardeingabe zu ermöglichen, ist es erforderlich, die Gleichungen und Ungleichungen des mathematischen Modells in ein Standardformat zu bringen.

```
+Z + a(0,1)*X1 + a(0,2)*X2 +...+ a(0,n)*Xn = RS(0)
--------------------------------------------------
   + a(1,1)*X1 + a(1,2)*X2 +...+ a(1,n)*Xn≤RS(1)
   + a(1,2)*X1 + a(2,2)*X2 +...+ a(2,n)*Xn≤RS(2)
   .
   .
   .
   + a(m,1)*X1 + a(m,2)*X2 +...+ a(m,n)*Xm≤RS(m)
```

Die Umformung ist so vorzunehmen, daß sämtliche Strukturvariablen X1 bis X2 sowie die Zielfunktionsvariable Z auf der linken Gleichungsseite stehen. Die Änderung der Vorzeichen ist dabei entsprechend zu berücksichtigen. Auf der rechten Seite RS verbleiben nur die absoluten Glieder (Glieder ohne Variable). Tritt bei den Restriktionen ein > oder ≥-Zeichen auf, so ist die gesamte Ungleichung mit -1 zu multiplizieren; aus der Zuordnung > wird dann <. Nun kann die Eintragung der Koeffizienten a(0,1) bis a(m,n) der Strukturvariablen sowie der rechten Seiten RS(0) bis RS(m) ins Simplex-Tableau erfolgen.

	X1	X2	....	Xn	RS
Z	a(0,1)	a(0,2)	....	a(0,n)	RS(0)
Y1	a(1,1)	a(1,2)	....	a(1,n)	RS(1)
Y2	a(2,1)	a(2,2)	....	a(2,n)	RS(2)
.					
.					
.					
Ym	a(m,1)	a(m,2)	....	a(m,n)	RS(m)

Programmbeschreibung

Das Programm Simplex 64 (Listing 1) ist mit LOAD "Name", 8 zu laden und mit RUN zu starten. Sämtliche für den Programmablauf notwendigen Eingaben erfolgen im Dialog mit dem C 64.

Nach dem Start von SIMPLEX 64 erscheint die Frage:
80 ZEICHENKARTE -(J/N)

Für die Bearbeitung von Optimierungsaufgaben mit mehr als zwei Strukturvariablen ist ein 80-Zeichen-Bildschirm aus Gründen der Übersichtlickeit vorteilhaft.

DRUCKERAUSGABE ? (J/N)
ALLE LÖSUNGEN? (1)
NUR OPTIMUM? (2)

Zwei unterschiedliche Druckoptionen stehen für die Ausgabe auf einem Drucker MPS-801 zur Verfügung. Bei der Wahl der Option (1) werden sämtliche Iterationen des Computers protokolliert. Sollen nur das Ausgangstableau und die optimale Lösung ausgedruckt werden, ist Option (2) zu wählen.

Die beiden folgenden Fragen
ANZAHL STRUKTURVARIABLEN?
ANZAHL RESTRIKTIONEN?
bedürfen kaum einer Erklärung; »Anzahl Restriktionen« bezieht sich auf die Anzahl der (Un-)Gleichungen ausschließlich der Zielfunktion.

Die nächsten Fragen betreffen die beschriebenen Sonderfälle »Gleichungen als Restriktionen« und »Freie Strukturvariablen«.

FREIE STRUKTURVARIABLEN? (J/N)
GESPERRTE SCHLUPFVARIABLEN? (J/N)

Enthält das Optimierungsproblem entsprechende Variablen, so sind im weiteren Verlauf die Spalten beziehungsweise Zeilen des Ausgangstableaus anzugeben, in denen diese Variablen auftreten.

Damit ist das Problem bereits weitgehend umschrieben - es fehlt nur noch die Eingabe der Koeffizienten und der RS (rechte Seite).

Auf dem Bildschirm erscheint die erste Zeile (Zielfunktion) eines leeren Simplex-Tableaus, oder, bei nicht ausreichender Bildschirmbreite, ein Teil derselben. Dieses Tableau ist nun vom Anwender entsprechend dem (hoffentlich) vorliegenden mathematischen Modell auszufüllen. Jede Eingabe eines Wertes ist mit <RETURN> abzuschließen.

Der C 64 beginnt mit der Berechnung, sobald der letzte Wert eingegeben wurde. Erscheint nach Beendigung der Iteration (Meldung: RECHNUNG BEENDET - OPTIMUM) nicht das vollständige Simplex-Tableau auf dem Bildschirm, so kann durch Drücken einer Taste der Rest abgefragt werden.

Interpretation des Ergebnis-Tableaus

Die Interpretation des Ergebnisses soll anhand des Beispiels 1 erläutert werden. Die Bilder 2 und 3 zeigen das Eingabe- (0. Iteration) und das Ausgabetableau. Auffällig ist, daß einige Variablen ihre Plätze getauscht haben. Die Variablen in der Spalte am linken Rand sind die Basisvariablen - sie stellen die Basis der Lösung dar. Die Variablen in der oberen Zeile sind Nichtbasisvariablen - sie haben den Wert Null.

Das abzulesende optimale Produktionsprogramm sieht also wie folgt aus:
Die herzustellende Menge der Produkte P1 (X1 = 5) und P2 (X2 = 60) ergeben sich zu 5 Stück vom P1 und 60 Stück von P2. Damit sind die Maschinen A (Y1 = 0) und C (Y3 = 0) 100prozentig ausgelastet. Für Maschine B verbleibt eine nicht genutzte Kapazität von 155 Stunden (Y2 = 155). Der Zielfunktionswert Z, also der Gewinn des Unternehmens, beläuft sich im betrachteten Zeitraum bei Durchführung des optimalen Produktionsprogramms auf 34800 Mark. Die Zielfunktionskoeffizienten (150 und 50) beziffern die sogeannten Opportunitätskosten der Maschinen A (Y1) und C (Y3). Jede Stunde Minderarbeit der Maschine A bedeutet eine Gewinnschmälerung von 150 Mark, jede Stunde Mehrarbeit einen betragsgleichen Gewinnzuwachs.

Natürlich steht jede Stunde Minder- oder Mehrarbeit der Maschine A im direkten Verhältnis zur Anzahl der hergestellten Produkte P1. Ein Beispiel soll diesen Zusammenhang verdeutlichen. Annahme: Maschine A fällt vier Stunden aus. Daraus folgt unmittelbar, daß die Nutzungskapazität der Maschine A (Y1) von 190 Stunden auf 186 Stunden sinkt. Wie sich dieser Ausfall auf die Anzahl der hergestellten Produkte P1 (X1) bemerkbar macht, läßt sich dem Lösungstableau (Bild 3) entnehmen. Dazu ist die Zeile hinter der Strukturvariablen X1 in eine Gleichung zu überführen:

0.5*Y1 - 0.5*Y3 = Anzahl der hergestellten Produkte P1
Y1, Y3 = Nutzungskapazitäten der Maschine A, C in Stunden

*** SIMPLEX TABLEAU ***

	X 1	X 2	RS
Z	-300	-600	-2700
Y 1	2	3	190
Y 2	3	0	170
Y 3	0	3	180

Bild 2. Eingabetableau von »Simplex 64«

```
     *** SIMPLEX TABLEAU  2.ITERATION ***

           Y 1       Y 3       RS

Y 0        150       50        34800

X 1        .5        -.5       5

Y 2        -1.5      1.5       155

X 2        0         .33       60

   **** RECHNUNG BEENDET ****
   ****      OPTIMUM     ****
```

Bild 3. Lösungstableau von »Simplex 64«

0.5, -0.5 = Multiplikationsfaktoren in Stück/Stunde
Daraus folgt:
0.5*186 - 0.5*180 = 3 = Anzahl der hergestellten Produkte P1 (X1). Bemerkenswert ist noch das negative Vorzeichen beim zweiten Multiplikationsfaktor. Fällt Maschine C (Y3) aus, so erhöht sich die Anzahl der hergestellten Produkte P1 (X1).
Allerdings nimmt P2 (X2) im gleichen Verhältnis ab.

Der vierstündige Ausfall von Maschine A (Y1) beeinflußt natürlich die nicht genutzte Stundenkapzität der Maschine B (Y2). Auch das läßt sich dem Lösungstableau (Bild 3) entnehmen:
Y2 - 1.5*Y1 + 1.5*Y3 = nicht genutzte Stundenkapazität der Maschine B (Y2)

Angewendet auf das oben stehende Beispiel ergibt sich die nicht genutzte Stundenkapazität zu:
170 - 1.5*186 + 1.5*180 = 161 Stunden.

Hinweise für interessierte Leser

Als Ergänzung zu dieser Einführung in die Simplex-Methode werde ich bei entsprechender Resonanz eine umfangreiche Beispielsammlung zusammenstellen.

(Manfred Buthz/ah)

```
100 REM ****************************        <091>
110 REM *                          *        <159>
120 REM *        SIMPLEX 64        *        <080>
130 REM *                          *        <179>
140 REM *            BY            *        <244>
150 REM *                          *        <199>
160 REM *         M. BUHTZ         *        <172>
170 REM *                          *        <219>
180 REM *      GRAFENRING 13       *        <195>
190 REM *        4230 WESEL        *        <027>
200 REM *      TEL.:0281/22431     *        <221>
210 REM *                          *        <003>
220 REM ****************************        <211>
230 :                                       <206>
240 PRINT"{CLR,5SPACE,RVSON}SIMPLEX 64{RVO
    FF}":PRINT:PRINT                        <169>
250 PRINT"{3SPACE}80 ZEICHENKARTE ? (J/N)"  <255>
260 GET A$                                  <216>
270 IF A$="J"THEN W=5:GOTO 300              <021>
280 IF A$="N"THEN W=1:GOTO 300              <225>
290 GOTO 260                                <068>
300 PRINT:PRINT:PRINT                       <171>
310 PRINT"{3SPACE}DRUCKERAUSGABE ? (J/N)":
    PRINT                                   <064>
320 GET A$                                  <020>
330 IF A$="N"THEN 410                       <152>
340 IF A$="J"THEN E1=1:GOTO 360             <228>
350 GOTO 320                                <072>
360 PRINT"{3SPACE}ALLE LOESUNGEN ? (1)":PR
    INT                                     <130>
370 PRINT"{3SPACE}NUR OPTIMUM ?{4SPACE}(2)
    "                                       <040>
380 GET E$                                  <112>
390 IF E$<>"1"AND E$<>"2"THEN 380           <221>
400 IF E$="2"THEN E1=2                      <163>
410 PRINT"{CLR,5SPACE,RVSON}SIMPLEX 64{RVO
    FF}":PRINT:PRINT:PRINT:PRINT            <184>
420 INPUT"{3SPACE}ANZAHL STRUKTURVARIABLEN
    ";N                                     <049>
430 PRINT                                   <022>
440 INPUT"{3SPACE}ANZAHL RESTRIKTIONEN ";M  <213>
450 M=M+1                                   <003>
460 PRINT"{CLR,4SPACE,RVSON}SIMPLEX 64{RVO
    FF}":PRINT:PRINT                        <234>
470 DIM A(M,N),B(M,N),E(M),X$(N),X1$(N),Y$
    (M),Y1$(M)                              <180>
480 PRINT"{3SPACE}FREIE STRUKTURVARIABLEN
    ? (J/N)"                                <062>
490 GET A$                                  <190>
500 IF A$="J"THEN DIM SP$(N) :GOTO 530      <247>
510 IF A$="N"THEN 610                       <084>
520 GOTO 490                                <108>
530 :                                       <254>
540 PRINT"{CLR,4SPACE,RVSON}FREIE STRUKTUR
    VARIABLEN{RVOFF}":PRINT                 <076>
550 FOR I=1 TO N                            <027>
560 PRINT"{DOWN,3SPACE}IN SPALTE"I"? (J/N)
    "                                       <131>
570 GET A$                                  <016>
580 IF A$=""THEN 570                        <031>
590 IF A$="J"THEN SP$(I)="J"                <108>
600 NEXT I                                  <176>
610 :                                       <078>
620 PRINT"{CLR,4SPACE,RVSON}SIMPLEX 64{RVO
    FF}":PRINT:PRINT:PRINT:PRINT            <140>
630 PRINT"{3SPACE}GESPERRTE SCHLUPFVARIABL
    EN ? (J/N)"                             <087>
640 GET A$                                  <086>
650 IF A$="J"THEN DIM Z$(M):GOTO 680        <127>
660 IF A$="N"THEN 770                       <032>
670 GOTO 640                                <194>
680 :                                       <148>
690 PRINT"{CLR,4SPACE,RVSON}GESPERRTE SCHL
    UPFVARIABLEN{RVOFF}":PRINT              <077>
700 FOR I=1 TO M-1                          <061>
710 PRINT"{DOWN,3SPACE}IN ZEILE"I"? (J/N)"  <159>
720 GET A$                                  <166>
730 IF A$=""THEN 720                        <165>
740 IF A$="J"THEN Z$(I)="J"                 <236>
750 NEXT I                                  <070>
760 :                                       <228>
770 REM *** EINGABE KOEFFIZIENTEN ***       <214>
780 :                                       <250>
790 FOR I1=1 TO N STEP W+1                  <217>
800 PRINT"{CLR}"                            <124>
810 PRINT,"*** SIMPLEX TABLEAU ***" :PRINT  <228>
820 X$(0)="RS "                             <011>
830 PRINT,                                  <011>
840 I2=I1+W                                 <080>
850 IF I2>=N THEN I2=N                      <118>
860 FOR I=I1 TO I2                          <081>
870 X$(I)="X":X1$(I)=STR$(I):X$(I)=X$(I)+X
    1$(I)                                   <124>
880 PRINT"X"I,                              <030>
890 NEXT I                                  <212>
900 IF I>N THEN PRINT"RS ",                 <129>
910 J1=J2+1                                 <204>
920 J2=J1+W                                 <196>
930 IF J2>=N THEN J2=N                      <208>
940 PRINT                                   <024>
950 FOR I=0 TO M-1                          <025>
960 Y$(I)="Y":Y1$(I)=STR$(I):Y$(I)=Y$(I)+Y
    1$(I)                                   <061>
970 IF I=0 THEN PRINT:PRINT;"{2SPACE}Z";:G
    OTO 1000                                <155>
980 IF I=1 THEN PRINT                       <088>
990 PRINT:PRINT;"Y"I;                       <239>
1000 FOR J=J1 TO J2                         <006>
1010 PRINT,;:POKE 19,1:INPUT A(I,J):POKE 1
     9,0                                    <242>
1020 NEXT J                                 <094>
1030 IF J>N THEN PRINT,"";:POKE 19,1:INPUT
      A(I,0):POKE 19,0                      <027>
1040 NEXT I                                 <108>
1050 NEXT I1                                <137>
1060 IF E1<>0 THEN GOSUB 2790               <109>
1070 :                                      <030>
1080 REM*** PHASE 0' S(SPALTE) ***          <225>
1090 :                                      <050>
1100 FOR I=1 TO N                           <069>
1110 IF SP$(I)="J"THEN S=I:SP$(I)="S":GOTO
     1170                                   <100>
1120 NEXT I                                 <188>
```

```
1130 GOTO 1220                                   <150>
1140 :                                           <100>
1150 REM*** PHASE 0' R(ZEILE) ***                <076>
1160 :                                           <120>
1170 R=INT(RND(1)*M)+1:T=T+1                     <059>
1180 IF Z$(R)="S"OR A(R,S)=0 THEN 1170           <074>
1190 IF T>100 THEN 2440                          <098>
1200 T=0:GOTO 1850                               <068>
1210 :                                           <170>
1220 REM*** PHASE 0 R(ZEILE) ***                 <009>
1230 :                                           <190>
1240 FOR I=1 TO M-1                              <093>
1250 IF Z$(I)="J"THEN R=I:Z$(I)="S":GOTO 1
     310                                         <056>
1260 NEXT I                                      <072>
1270 GOTO 1360                                   <178>
1280 :                                           <242>
1290 REM*** PHASE 0 S(SPALTE) ***                <063>
1300 :                                           <006>
1310 S=INT(RND(1)*N)+1:T=T+1                     <213>
1320 IF SP$(S)="S"OR A(R,S)=0 THEN 1310          <201>
1330 IF T>100 THEN 2530                          <236>
1340 T=0:GOTO 1850                               <210>
1350 :                                           <056>
1360 REM *** PHASE 1 R(ZEILE) ***                <159>
1370 :                                           <076>
1380 S=0:R=0                                     <183>
1390 FOR I=1 TO M-1                              <245>
1400 IF A(I,0)<0 THEN R=I:GOTO 1450              <102>
1410 NEXT I:GOTO 1520                            <044>
1420 :                                           <126>
1430 REM*** PHASE 1 S(SPALTE) ***                <211>
1440 :                                           <146>
1450 S=INT(RND(1)*N)+1:T=T+1                     <097>
1460 IF T>100 THEN 2620                          <106>
1470 IF A(R,S)>=0 THEN 1450                      <206>
1480 IF R<>0 THEN T=0:GOTO 1850                  <194>
1490 :                                           <196>
1500 REM *** PHASE 2 S(SPALTE) ***               <033>
1510 :                                           <216>
1520 S=0:R=0                                     <067>
1530 IF A(0,1)<0 THEN S=1                        <203>
1540 FOR J=2 TO N                                <041>
1550 IF A(0,J)>=0 OR  A(0,J)>A(0,J-1) THEN
     1570                                        <200>
1560 S=J                                         <170>
1570 NEXT J                                      <138>
1580 IF S>0 THEN 1710                            <251>
1590 EE=1                                        <210>
1600 IF E1=2 THEN GOSUB 2800                     <193>
1610 PRINT                                       <188>
1620 PRINT" **** RECHNUNG BEENDET ****"          <236>
1630 PRINT" ****{6SPACE}OPTIMUM{5SPACE}***
     *"                                          <014>
1640 IF E1=0 THEN 1670                           <014>
1650 OPEN 4,4:PRINT#4:PRINT#4,CHR$(16)"11*
     *** RECHNUNG BEENDET ****"                  <166>
1660 PRINT#4,CHR$(16)"11****{6SPACE}OPTIMU
     M{5SPACE}****":PRINT#4:CLOSE 4              <197>
1670 GET A$                                      <100>
1680 IF A$<>""THEN 2070                          <060>
1690 GOTO 1670                                   <170>
1700 :                                           <152>
1710 REM *** PHASE 2 R(ZEILE) ***                <007>
1720 :                                           <172>
1730 FOR I=1 TO M-1                              <075>
1740 IF A(I,S)<=0 THEN E(I)=10↑38:GOTO 176
     0                                           <227>
1750 E(I)=A(I,0)/A(I,S)                          <010>
1760 NEXT I                                      <064>
1770 IF E(1)>0 THEN R=1:GOTO 1790                <050>
1780 E(1)=10↑38                                  <236>
1790 FOR I=2 TO M-1                              <167>
1800 IF E(I)<=0 OR E(I)>=E(I-1)THEN 1820         <020>
1810 R=I:GOTO 1850                               <053>
1820 NEXT I                                      <126>
1830 IF R<>1 THEN 2710                           <105>
1840 :                                           <038>
1850 REM *** UMRECHNUNG ***                      <206>
1860 :                                           <058>
1870 ZW$=X$(S):X$(S)=Y$(R):Y$(R)=ZW$             <165>
1880 B(R,S)=1/A(R,S)                             <063>
1890 :                                           <088>
1900 FOR J=0 TO N                                <083>
1910 IF J=S THEN 1930                            <121>
1920 B(R,J)=A(R,J)/A(R,S)                        <206>
1930 NEXT J                                      <244>
1940 :                                           <138>
1950 FOR I=0 TO M-1                              <009>
1960 IF I=R THEN 1980                            <141>
1970 B(I,S)=-A(I,S)/A(R,S)                       <050>
1980 NEXT I                                      <030>
1990 :                                           <188>
2000 FOR I=0 TO M-1                              <059>
2010 FOR J=0 TO N                                <193>
2020 IF J=S OR I=R THEN 2040                     <241>
2030 B(I,J)=A(I,J)-A(I,S)*B(R,J)                 <234>
2040 NEXT J                                      <098>
2050 NEXT I                                      <102>
2060 :                                           <004>
2070 REM *** UMGERECHNETES SIMPLEX TABLEAU
      ***                                        <025>
2080 :                                           <124>
2090 IT=IT+1                                     <240>
2100 I1=0:I2=0:J1=0:J2=0                         <057>
2110 FOR I1=1 TO N STEP W+1                      <011>
2120 IF N<=W+1 THEN 2130                         <189>
2130 PRINT"{CLR}"                                <086>
2140 PRINT"{3SPACE}*** SIMPLEX TABLEAU "IT
     "{LEFT}.ITERATION ***":PRINT                <169>
2150 PRINT,                                      <061>
2160 I2=I1+W                                     <130>
2170 IF I2=>N THEN I2=N                          <136>
2180 FOR I=I1 TO I2                              <131>
2190 PRINT X$(I),                                <031>
2200 NEXT I                                      <252>
2210 IF I>N THEN PRINT X$(0)                     <022>
2220 J1=J2+1                                     <244>
2230 J2=J1+W                                     <236>
2240 IF J2>=N THEN J2=N                          <248>
2250 FOR I=0 TO M-1                              <055>
2260 PRINT:PRINT Y$(I);                          <131>
2270 FOR J=J1 TO J2                              <006>
2280 A(I,J)=B(I,J)                               <242>
2290 A(I,0)=B(I,0)                               <051>
2300 PRINT,"";:PRINT INT(A(I,J)*100+.5)/10
     0;                                          <036>
2310 NEXT J                                      <116>
2320 IF J>N THEN PRINT,"";:PRINT INT(A(I,0
     )*100+.5)/100                               <027>
2330 NEXT I                                      <128>
2340 IF EE<>1 OR I1>=N THEN 2370                 <070>
2350 GET A$                                      <018>
2360 IF A$=""THEN 2350                           <054>
2370 NEXT I1                                     <187>
2380 :                                           <070>
2390 IF E1=1 THEN GOSUB 2790                     <225>
2400 GOTO 1070                                   <022>
2410 :                                           <100>
2420 REM *** MELDUNGEN ***                       <206>
2430 :                                           <120>
2440 PRINT:PRINT" *** RECHNUNG ABGEBROCHEN
      PHASE 0' ***"                              <088>
2450 PRINT" ***{3SPACE}LINEARKOMBINATION S
     PALTEN{3SPACE}***"                          <049>
2460 IF E1=2 THEN GOSUB 2800                     <035>
2470 IF E1=0 THEN END                            <216>
2480 OPEN 4,4:PRINT#4                            <055>
2490 PRINT#4,CHR$(16)"11*** RECHNUNG ABGEB
     ROCHEN PHASE 0' ***"                        <164>
2500 PRINT#4,CHR$(16)"11***{3SPACE}LINEARK
     OMBINATION SPALTEN{3SPACE}***"              <024>
2510 PRINT#4:CLOSE 4:END                         <068>
2520 :                                           <210>
2530 PRINT:PRINT" *** RECHNUNG ABGEBROCHEN
      PHASE 0 ***"                               <053>
2540 PRINT" ***{3SPACE}LINEARKOMBINATION Z
     EILEN{3SPACE}***"                           <043>
2550 IF E1=2 THEN GOSUB 2800                     <125>
2560 IF E1=0 THEN END                            <052>
2570 OPEN 4,4:PRINT#4                            <147>
2580 PRINT#4,CHR$(16)"11*** RECHNUNG ABGEB
     ROCHEN PHASE 0 ***"                         <066>
2590 PRINT#4,CHR$(16)"11***{3SPACE}LINEARK
     OMBINATION ZEILEN{3SPACE}***"               <179>
2600 PRINT#4:CLOSE 4:END                         <160>
2610 :                                           <046>
2620 PRINT:PRINT" *** RECHNUNG ABGEBROCHEN
      PHASE 1 ***"                               <147>
2630 PRINT" ***{3SPACE}KEINE ZULAESSIGE LO
     ESUNG{3SPACE}***"                           <167>
2640 IF E1=2 THEN GOSUB 2800                     <217>
2650 IF E1=0 THEN END                            <142>
```

Listing 1. »Simplex 64«, ein Programm zur Optimierung linearer Zusammenhänge

```
2660 OPEN 4,4:PRINT#4                        <237>
2670 PRINT#4,CHR$(16)"11*** RECHNUNG ABGEB
     ROCHEN PHASE 1 ***"                      <157>
2680 PRINT#4,CHR$(16)"11***{3SPACE}KEINE Z
     ULAESSIGE LOESUNG{3SPACE}***":PRINT#4
     :CLOSE 4:END                            <002>
2690 PRINT#4:CLOSE 4:END                     <250>
2700 :                                       <136>
2710 PRINT:PRINT" *** RECHNUNG ABGEBROCHEN
      PHASE 2 ***"                           <239>
2720 PRINT" ***{3SPACE}KEINE BEGRENZTE LOE
     SUNG{4SPACE}***"                        <113>
2730 IF E1=2 THEN GOSUB 2800                 <051>
2740 IF E1=0 THEN END                        <232>
2750 OPEN 4,4:PRINT#4                        <071>
2760 PRINT#4,CHR$(16)"11*** RECHNUNG ABGEB
     ROCHEN PHASE 2 ***"                      <248>
2770 PRINT#4,CHR$(16)"11***{3SPACE}KEINE B
     EGRENZTE LOESUNG{4SPACE}***":PRINT#4:
     CLOSE 4:END                             <099>
2780 PRINT#4:CLOSE 4:END                     <084>
2790 :                                       <226>
2800 REM *** DRUCKERAUSGABE ***              <158>
2810 :                                       <246>
2820 OPEN 4,4                                <129>
2830 I2=0:J1=0:J2=0                          <147>
2840 FOR I1=1 TO N STEP 4                    <246>
2850 PRINT#4,CHR$(10)                        <025>
2860 PRINT#4,CHR$(16)"10*** SIMPLEX TABLEA
     U"IT".ITERATION ***"                    <069>
2870 PRINT#4                                 <079>
2880 I2=I1+3                                 <052>
2890 IF I2>=N THEN I2=N                      <126>
2900 Z=0                                     <107>
2910 FOR I=I1 TO I2                          <099>
2920 Z=Z+1                                   <194>
2930 ON Z GOSUB 3260,3270,3280,3290,3300     <011>
2940 :                                       <122>
2950 NEXT I                                  <240>
2960 IF I<=N THEN 2990                       <030>
2970 I=0:Z=Z+1                               <022>
2980 ON Z GOSUB 3260,3270,3280,3290,3300     <061>
2990 J1=J2+1                                 <252>
3000 J2=J1+3                                 <208>
3010 IF J2>=N THEN J2=N                      <000>
3020 PRINT#4                                 <229>
3030 FOR I=0 TO M-1                          <073>
3040 IF I=0 THEN PRINT#4:PRINT#4,"{2SPACE}
     Z",:GOTO 3070                           <128>
3050 IF I=1 THEN PRINT#4                     <050>
3060 PRINT#4:PRINT#4,Y$(I),                  <136>
3070 Z=0                                     <021>
3080 FOR J=J1 TO J2                          <056>
3090 IF IT=0 THEN 3120                       <250>
3100 A(I,J)=B(I,J)                           <046>
3110 A(I,0)=B(I,0)                           <111>
3120 Z=Z+1                                   <140>
3130 ON Z GOSUB 3210,3220,3230,3240,3250     <219>
3140 NEXT J                                  <184>
3150 IF J<=N THEN 3180                       <211>
3160 J=0:Z=Z+1                               <218>
3170 ON Z GOSUB 3210,3220,3230,3240,3250     <003>
3180 NEXT I                                  <216>
3190 NEXT I1                                 <245>
3200 PRINT#4:CLOSE 4:RETURN                  <010>
3210 PRINT#4,CHR$(16)"13"INT(A(I,J)*100+.5
     )/100;:RETURN                           <144>
3220 PRINT#4,CHR$(16)"26"INT(A(I,J)*100+.5
     )/100;:RETURN                           <210>
3230 PRINT#4,CHR$(16)"39"INT(A(I,J)*100+.5
     )/100;:RETURN                           <020>
3240 PRINT#4,CHR$(16)"52"INT(A(I,J)*100+.5
     )/100;:RETURN                           <190>
3250 PRINT#4,CHR$(16)"65"INT(A(I,J)*100+.5
     )/100;:RETURN                           <000>
3260 PRINT#4,CHR$(16)"14"X$(I);:RETURN       <038>
3270 PRINT#4,CHR$(16)"27"X$(I);:RETURN       <104>
3280 PRINT#4,CHR$(16)"40"X$(I);:RETURN       <018>
3290 PRINT#4,CHR$(16)"53"X$(I);:RETURN       <084>
3300 PRINT#4,CHR$(16)"66"X$(I);:RETURN       <150>
```

Listing 1. »Simplex 64« (Schluß)

Polydat – Dateiverwaltung schnell und komfortabel

Wer viel mit Daten, zum Beispiel Adressen, Bücher-, Software- und Autorenlisten oder ähnlichem arbeitet, benötigt ein leistungsfähiges und komfortables Dateiverwaltungsprogramm. Mit »Polydat« wird universelle Datenverarbeitung zum reinsten Vergnügen.

Dateiverwaltung ist eines der bekanntesten und beliebtesten Anwendungsgebiete eines Computers. Gibt es doch eine Menge Wissenswertes im Leben eines Menschen, das man sich merken muß und das jederzeit abrufbar sein soll. Und hier zeigt der Computer mit der entsprechenden Software seine wahre Stärke. Es wäre jedoch unnötiger Aufwand, wollte man für jede spezielle Anwendung ein eigenes Programm entwickeln. Wozu gibt es schließlich universelle Dateiverwaltungsprogramme? Universell heißt, daß Sie, egal was immer Sie verwalten wollen – ob Adressen, Briefmarken-, Münzen- oder andere Sammlungen – für alle Dinge ein und dasselbe Programm verwenden können. Die Ein- und Ausgabemaske kann also bei einem derartigen Programm ganz nach Belieben und Bedarf entworfen und geändert werden. Polydat (Listing 1 und 2) ist ein solches universell einsetzbares Dateiverwaltungsprogramm, das einen sehr schnellen Zugriff auf die gespeicherten Daten über ein frei wählbares Schlüsselfeld erlaubt. Sie können also selbst entscheiden, über welches Feld Ihrer Eingabemaske ein direkter Zugriff möglich sein soll. Die Suche nach anderen Feldern dauert natürlich etwas länger, da in diesem Fall auf der Diskette gesucht werden muß.

Komfortable Menüsteuerung

Das besonders interessante an Polydat ist, daß es sich nahezu selbst erklärt: Alle Funktionen des Programms sind als eindeutige Menüpunkte ausgewiesen und werden über entsprechende Tasten ausgeführt. Fehleingaben sind nahezu ausgeschlossen oder werden ignoriert. An jeder beliebigen Stelle kann mit <F2> ins Hauptmenü zurückgekehrt werden, ohne daß Daten verloren gehen (geöffnete Dateien werden geschlossen). Bei allen Eingaben besteht die Möglichkeit, den Cursor mit den Cursortasten innerhalb

des Eingabefeldes zu bewegen und die Eingabe mit den Tasten <INST> und <DEL> zu editieren. Mit <SHIFT CLR/HOME> kann das gesamte Eingabefeld gelöscht werden. Unerlaubte Zeichen, wie zum Beispiel Steuerzeichen für Farbänderungen, Bildschirmlöschen oder ähnliches werden einfach ignoriert. Es ist also ungeheuer schwer, Fehler zu machen.

Das Hauptmenü beinhaltet sechs Punkte, die durch Druck auf die entsprechende Zifferntaste erreicht werden (Bild 1). Diese Menüpunkte sind hierarchisch aufgebaut, das heißt, die Reihenfolge der Menüpunkte entspricht Ihrer Vorgehensweise beim Arbeiten. Kommen wir nun zur Beschreibung der einzelnen Programmfunktionen.

Datei einrichten

Nach Aufruf dieses Menüpunktes wird abgefragt, ob eine bereits vorhandene Eingabemaske verwendet werden soll. Geben Sie hier »J« für Ja ein, werden Sie aufgefordert, den Namen dieser Maske einzugeben. Über eventuell auftretende Fehler beim Ladevorgang werden Sie vom Programm informiert. Falls keine fertige Maske verwendet werden soll, können Sie nun eine neue Eingabemaske erstellen. Im Masken-Editor befindet sich der Computer im Groß-/Kleinschrift-Modus, wobei der komplette Zeichensatz zur Verfügung steht. Zur Vereinfachung der Eingabe wurde die Repeatfunktion, die sonst nur von <SPACE> und den Cursortasten her bekannt ist, auf alle Tasten erweitert. Das Cursorblinken wurde zur besseren Orientierung abgestellt. Folgende Tasten sind im Editor mit Funktionen belegt:

<F1> Beenden der Editierfunktion
<F2> Abbruch (Rücksprung ins Hauptmenü)
<F3> Zeile löschen
<F4> Zeile einfügen

Eingabefelder werden bei Polydat mit <@> gekennzeichnet. Für jede Stelle eines Eingabefeldes muß dieses Zeichen gesetzt werden, zum Beispiel:

```
Name : @@@@@@@@@@@@@@@@@@@@@@@@
```

Eine Datei darf maximal 30 Eingabefelder mit einer gesamten Feldlänge von 254 Zeichen besitzen. Über fehlerhafte Eingaben bei der Maske wird der Anwender vom Programm informiert.

Nach dem Laden oder dem Editieren einer Maske besteht nun die Möglichkeit, diese gegebenenfalls zu verändern. Ist die Maske fertiggestellt, fragt das Programm nach dem Indexfeld. Als Indexfeld sollte das Eingabefeld gewählt werden, das später beim Suchen besonders häufig verwendet wird, da die Suche über ein Indexfeld besonders schnell durchgeführt werden kann.

Danach werden Sie aufgefordert, eine Datendiskette einzulegen und eine Taste zu drücken. An dieser Stelle muß der neuen Datei ein Name gegeben werden, worauf die Maske und das Indexfeld gespeichert und eine relative Datei eingerichtet wird. Danach können Sie sofort mit der Dateneingabe beginnen.

Daten eingeben

In diesem Programmteil können Daten eingegeben und gespeichert werden. Nach der Eingabe erscheint die in Menüpunkt 1 editierte Eingabemaske auf dem Monitor. Dort werden kurz die wichtigsten Befehle angezeigt. Zur Eingabe sind folgende Funktionen vorgesehen:

<RETURN> : Sprung ins nächste Eingabefeld
<SHIFT RETURN> : Sprung ins vorhergehende Eingabefeld
<F1> : Beenden der Eingabe
<F2> : Abbruch und Rücksprung zum Hauptmenü

Nach Beenden der Eingabe mit <F1> können weiterhin folgende Funktionen aufgerufen werden:

<E> : Beenden der Dateneingabe und Schließen der Datei
<A> : Ändern des angezeigten Datensatzes
<F1> : Eingabe weiterer Datensätze
<F7> : In der Kommandozeile wird angezeigt, wie viele Datensätze bereits eingegeben wurden und wie viele noch eingegeben werden können

Daten suchen

Auch hier wird zuerst der Name der Datei eingegeben. Dann fragt das Programm, ob nach einzelnen Datensätzen oder ganzen Blöcken gesucht werden soll. Wird nach einem bestimmten Bereich gesucht, so besteht die Möglichkeit, sich alle Datensätze oder auch nur einzelne ausdrucken zu lassen.

Nach der Eingabe des Suchkriteriums beginnt der Computer mit der Suche und gibt die gewünschten Datensätze je nach Wahl auf Bildschirm oder Drucker aus. Das Suchen nach einzelnen Datensätzen verläuft ähnlich. Allerdings wird das Suchen nach dem ersten gefundenen Datensatz unterbrochen. Hier können nun folgende Funktionen ausgeführt werden:

<D> : Ausdruck des angezeigten Bildschirms
<A> : Ändern des angezeigten Datensatzes
<+>/
<-> : Anzeigen des nachfolgenden/vorhergehenden Datensatzes

Nach Drücken einer anderen Taste sind folgende Unterpunkte erreichbar:

<W> : Der Suchvorgang wird fortgesetzt
<N> : Das Suchkriterium kann geändert werden
<F1> : Die Suche wird beendet

Die Eingabe der Suchkriterien erfolgt in allen Eingabefeldern gleich, jedoch ist die Suche nach Kriterien im Indexfeld erheblich schneller. Eine Eingabe kann mit <*> abgekürzt werden. Einzelne Zeichen können durch ein <?> ersetzt werden, wenn sie nicht bekannt oder für die Suche irrelevant sind.

Druckerparameter

Da die meisten Drucker im Textmodus gleich angesteuert werden, ist keine spezielle Anpassung für die Druckerroutine notwendig. In diesem Menüpunkt kann zwischen Normalschrift und Breitschrift gewählt werden. Da bei Normalschrift nur die Hälfte des Papiers ausgenutzt wird, kann hier weiterhin der Abstand vom linken Papierrand bestimmt werden.

Falls die Druckerausgabe nicht funktionieren sollte, verwenden Sie bitte das Programm »Druckeranpassung« (Listing 3), mit dem die Geräteadresse, die Sekundäradresse und der Steuercode für Breitschrift ins Programm »D2« eingebunden werden können. Voreingestellt sind Geräteadresse #4, Sekundäradresse #7 und #14 für Breitschrift einschalten.

```
H A U P T M E N U E

( 1 )   DATEI EINRICHTEN
( 2 )   DATEN EINGEBEN
( 3 )   DATEN SUCHEN
( 4 )   DRUCKERPARAMETER
( 5 )   DISKOPERATIONEN
( 6 )   PROGRAMM ENDE
```

Bild 1. Das Hauptmenü von »Polydat« ist durch die Reihenfolge der einzelnen Menüpunkte logisch aufgebaut

Diskoperationen

Die im folgenden aufgeführten Funktionen werden vom Programm ausführlich erklärt und benötigen keine weitere Beschreibung.

<1> Directory (Inhaltsverzeichnis der Diskette)
<2> Datei löschen
<3> Datei umbenennen
<4> Disk formatieren
<5> Disk validieren
<6> Hauptmenü

Lediglich bei den Punkten »Datei löschen« und »Datei umbenennen« ist zu beachten, daß diese Befehle gleichzeitig für die Masken-, Index- und Datenfiles gelten. Es werden also jeweils alle drei Dateien umbenannt oder gelöscht. Weiterhin sind von diesen Funktionen auch andere Programme betroffen, die den eingegebenen Dateinamen tragen.

Sicher haben auch Sie bald eine Sammlung von Daten. Mit der unübersichtlichen Zettelwirtschaft ist dann aber endgültig Schluß. (C. Süßkind, M. Kereit/nj)

```
70 REM **********************************     <120>
80 REM *          INITIALISIEREN          *   <095>
90 REM **********************************     <140>
100 POKE 55,0:POKE 56,58:CLR                 <220>
110 PRINT"{CLR}":POKE 808,254                 <106>
120 VI=53248:POKE VI+32,6:POKE VI+33,6        <001>
130 FOR I=0 TO 62:READ X:POKE 832+I,X         <155>
140 NEXT I:POKE 2040,13:POKE VI,55            <142>
150 POKE VI+1,195:POKE 657,128                <119>
160 S$="********************"                 <210>
170 T$=S$+"****************"                  <113>
180 U$="TTTTTTTTTTTTTTTTT"                    <025>
190 POKE 3,0:POKE 4,0                         <034>
200 PRINT CHR$(142)"{CLR,2DOWN,YELLOW}"TAB
    (9)"U"S$"I":PRINT TAB(9)"-{4SPACE}POLY
    DAT V2.0{4SPACE}-"                        <101>
210 PRINT TAB(9)"J"S$"K":PRINT"{DOWN,SPACE
    }U"T$"I{2SPACE}-"SPC(36)"-{2SPACE}-{2S
    PACE}AUTOREN :";                          <102>
220 PRINT SPC(25)"-{2SPACE}-"SPC(36)"-{2SP
    ACE}-{8SPACE}CHRISTIAN SUESSKIND{9SPAC
    E}-"                                      <232>
230 PRINT" -"SPC(36)"-{2SPACE}-{8SPACE}MAT
    THIAS KEREIT{13SPACE}-{2SPACE}-"SPC(36
    )"-"                                      <130>
240 PRINT" J"T$"K{DOWN,2SPACE}U"T$"I"         <252>
250 PRINT" -"SPC(36)"-{2SPACE}-{8SPACE}COP
    YRIGHT 1987{14SPACE}-{2SPACE}-"SPC(36)
    "- ";                                     <244>
260 PRINT" -{8SPACE}BY MARKT & TECHNIK{10S
    PACE}-{2SPACE}-"SPC(36)"-{2SPACE}J"T$"
    K{3UP}"                                   <056>
270 POKE VI+21,1:IF PEEK(49152)=120 AND PE
    EK(49153)=165 AND PEEK(49154)=1 THEN 3
    10                                        <174>
280 SYS(57812)"D2",8:POKE 780,0:POKE 781,0
    :POKE 782,192:SYS 65493                   <225>
290 OPEN 15,8,15:INPUT#15,A,B$,C,D:CLOSE 1
    5:IF A=0 THEN 310                         <087>
300 PRINT"{2DOWN,6SPACE}BITTE PROGRAMMDISK
     EINLEGEN !":GOSUB 8100:GOTO 200          <199>
310 GOSUB 8600:POKE VI+21,0                   <234>
370 ID=15000:DIM X(30),Y(30),L(30)            <198>
380 PRINT"{CLR}":GOSUB 8900                   <177>
760 :                                         <228>
770 REM **********************************    <146>
780 REM *           HAUPT-MENUE          *    <203>
790 REM **********************************    <166>
800 SYS 52258                                 <124>
810 GOSUB 8100:A=ASC(A$)-48                   <198>
820 IF A<1 OR A>6 THEN 810                    <052>
830 GOSUB 8700:ON A GOTO 1000,2000,3000,40
    00,5000,6000                              <174>
960 :                                         <174>
970 REM **********************************    <090>
980 REM *        DATEI EINRICHTEN        *    <160>
990 REM **********************************    <110>
1000 A$="DATEI EINRICHTEN":GOSUB 9000:PRIN
     T"{DOWN,3SPACE}ALTE MASKE VERWENDEN (
     J/N) ?                                   <042>
1010 GOSUB 8200:IF A$="J"THEN 1900            <029>
1015 GOSUB 8800:IF DN$<>""THEN PRINT"{CLR}
     "                                        <061>
1020 DN$="":SYS 51171                         <207>
1030 KO$="{6SPACE}'F1' ENDE{5SPACE}'F2' AB
     BRUCH{6SPACE}":GOSUB 9700                <153>
1040 POKE 250,0:SYS 50052:IF PEEK(250)<>14
     7 THEN 1050                              <112>
1045 GOSUB 8100:IF A$="{CLR}"THEN PRINT A$ <034>
1048 GOTO 1020                                <036>
1050 IF PEEK(250)=137 THEN GOSUB 8900:GOTO
      9600                                    <034>
1060 GOSUB 8900:SYS 49210:GOSUB 9210:IF AE
      AND AE<=30 AND RL<255 THEN 1400         <035>
1070 IF AE=0 THEN KO$="{5SPACE}KEIN EINGAB
     EFELD VORHANDEN !{5SPACE}"               <144>
1080 IF AE>30 THEN KO$="{7SPACE}ZUVIELE EI
     NGABEFELDER !{8SPACE}"                   <158>
1090 IF RL>254 THEN KO$="{10SPACE}DATENSAT
     Z ZU LANG !{9SPACE}"                     <231>
1100 GOSUB 9700:GOSUB 8100:GOTO 1030          <209>
1400 A$="DATEI EINRICHTEN":GOSUB 9000         <046>
1410 PRINT CHR$(142)"{DOWN,3SPACE}MASKE AE
     NDERN (J/N) ?":GOSUB 8200                <008>
1420 IF A$="J"THEN GOSUB 8800:GOTO 1030       <116>
1500 IF AE=1 THEN ZA=1:GOTO 1530              <099>
1510 PRINT"{DOWN,3SPACE}INDEXFELD (1";STR$
     (-AE);") :";:L=LEN(STR$(AE))-1           <050>
1520 GOSUB 8300:IF ZA<1 OR ZA>AE OR A=133
     THEN PRINT"{3UP}":GOTO 1510              <059>
1530 IN=ZA:POKE ID+1,IN:PRINT                 <139>
1540 PRINT"{3SPACE}DATENDISK EINLEGEN !{6S
     PACE}>TASTE<":GOSUB 8100                 <138>
1560 PRINT"{UP,36SPACE,2UP}":OPEN 15,8,15:
     GOSUB 9210                               <170>
1570 GOSUB 9800:CLOSE 15:FB=FB-6              <114>
1680 IF A THEN GOSUB 8530:GOTO 1550           <140>
1690 IF FB<0 THEN PRINT"{DOWN,3SPACE}DISKE
     TTE VOLL !{6SPACE}>TASTE<{UP}":GOSUB
     8100:GOTO 1540                           <146>
1700 NE=1:POKE ID+4,1:POKE ID+5,0             <006>
1720 GOSUB 9300:DN$=NA$                       <063>
1730 SYS(57812)"M-"+DN$,8                     <199>
1740 SYS 52926:OPEN 15,8,15                   <115>
1750 INPUT#15,A,B$,C,D:IF A=0 THEN 1790       <174>
1760 IF A<>63 THEN GOSUB 8530:GOTO 1730       <125>
1770 PRINT"{DOWN,3SPACE}DATEI BEREITS VORH
     ANDEN !":PRINT"{DOWN,3SPACE}UEBERSCHR
     EIBEN (J/N) ?"                           <220>
1780 GOSUB 8200:IF A$="N"THEN CLOSE 15:GOT
     O 1720                                   <060>
1785 PRINT#15,"S:??"DN$:CLOSE 15:GOTO 1730 <236>
1790 CLOSE 15:SYS(57812)"I-"+DN$,8            <182>
1800 POKE 193,152:POKE 194,58                 <216>
1810 X=ID+7+3*AE+NU*(IL+2):POKE 175,X/256:
     POKE 174,X-256*PEEK(175):SYS 62957       <102>
1820 GOSUB 8500:IF A THEN 1790                <125>
1830 OPEN 1,8,2,"{2SPACE}"+DN$+",L,"+CHR$(
     RL)                                      <246>
1840 CLOSE 1                                  <073>
1860 PRINT"{DOWN,3SPACE}DATEN EINGEBEN (J/
     N) ?":GOSUB 8200                         <147>
1870 IF A$="J"THEN GOSUB 9200:GOTO 2020       <220>
1880 GOTO 800                                 <086>
1900 PRINT"{DOWN,3SPACE}NAME DER MASKE :"; <196>
1910 NA$=DN$:L=14:GOSUB 8400                  <195>
1920 SYS(57812)"M-"+NA$,8:POKE 780,0:POKE
     781,0:POKE 782,160:SYS 65493             <074>
1930 GOSUB 8500:IF A THEN 1000                <090>
1940 SYS 49210:AE=PEEK(ID):GOTO 1410          <198>
1960 :                                        <158>
1970 REM ********************************     <246>
1980 REM *        DATEN EINGEBEN        *     <219>
1990 REM ********************************     <010>
2000 A$="DATEN EINGEBEN":GOSUB 9000:GOSUB
     9300                                     <006>
```

Listing 1. Basic-Hauptprogramm »Polydat«. Bitte beachten Sie die Eingabehinweise auf Seite 159.

```
2010 GOSUB 9100:IF A THEN 2000                  <122>
2020 AL=NE:OPEN 15,8,15:GOSUB 9800              <226>
2100 CLOSE 15:FB=INT(252*FB/RL):FB=INT(FB/
     (1+IL/254))                                <122>
2105 IF FB<1 THEN GOSUB 8800:GOTO 2460          <229>
2110 OPEN 1,8,2,"{2SPACE}"+DN$:GOSUB 8510       <031>
2112 IF A THEN CLOSE 1:GOTO 2000                <176>
2115 GOSUB 8800                                 <223>
2120 X=INT(NE/256):Y=NE-256*X:PRINT#15,"P"
     +CHR$(2)+CHR$(Y)+CHR$(X)+CHR$(1)           <196>
2125 INPUT#15,A,B$,C,D                          <212>
2130 KO$=" 'F1' ENDE DER EINGABE{2SPACE}'F
     2' ABBRUCH{2SPACE}":I=1                    <026>
2140 GOSUB 9700:GOSUB 9510:SYS 49979            <196>
2300 IE=2:X=ID+6+3*AE+(NE-1)*(IL+2):POKE X
     +1,NE/256:POKE X,NE-256*PEEK(X+1)          <015>
2310 FOR I=1 TO L(IN):POKE X+1+I,PEEK(1023
     +40*Y(IN)+X(IN)+I):NEXT I:NE=NE+1          <002>
2400 KO$=" 'E' ENDE{2SPACE}'A' AENDERN{2SP
     ACE}'F1' EINGEBEN ":GOSUB 9700             <225>
2410 GOSUB 8100:IF A$="A"THEN NE=NE-1:IE=3
     :GOTO 2120                                 <146>
2420 IF A$="E"THEN 2500                         <107>
2430 IF A$="{F7}"THEN CLOSE 1:CLOSE 15:GOS
     UB 9900:GOSUB 2800:GOTO 2400               <145>
2440 IF A$<>"{F1}"THEN 2410                     <116>
2450 FB=FB-1:IF FB>0 THEN 2120                  <066>
2460 CLOSE 1:CLOSE 15                           <086>
2470 KO$=" ENDE DER EINGABE !{3SPACE}DISKE
     TTE VOLL ! ":GOSUB 9700                    <158>
2480 IF IE=0 THEN GOSUB 8100:GOTO 800           <093>
2500 POKE ID+5,NE/256:POKE ID+4,NE-256*PEE
     K(ID+5):IF NE=AL THEN 2700                 <151>
2510 POKE 248,AL/256:POKE 247,AL-256*PEEK(
     248):POKE 189,IL+2:SYS 49514               <080>
2700 CLOSE 1:CLOSE 15                           <072>
2710 OPEN 1,8,15,"S:I-"+DN$:CLOSE 1             <131>
2720 SYS(57812)"I-"+DN$,8:POKE 193,152          <037>
2730 POKE 194,58:X=ID+7+3*AE+NE*(IL+2)          <205>
2740 POKE 175,X/256:POKE 174,X-256*PEEK(17
     5):SYS 62957:IE=0:GOTO 800                 <155>
2800 OPEN 1,8,2,"{2SPACE}"+DN$:GOTO 8510        <209>
2960 :                                          <142>
2970 REM ******************************         <230>
2980 REM *          DATEN SUCHEN       *        <163>
2990 REM ******************************         <250>
3000 GF=0:IE=0:A$="DATEN SUCHEN":GOSUB 900
     0:GOSUB 9300                               <108>
3010 GOSUB 9100:IF A THEN 3000                  <114>
3020 IF NE=1 THEN PRINT"{DOWN,3SPACE}KEINE
      DATEN VORHANDEN !":GOSUB 8100:GOTO 8
     00                                         <111>
3021 PRINT"{DOWN,3SPACE}BEREICH ODER EINZE
     LN SUCHEN (B/E) ?"                         <030>
3022 GOSUB 8100:IF A$<>"B"AND A$<>"E"THEN
     3022                                       <247>
3023 SU$=A$:IF SU$="E"THEN 3025                 <103>
3024 PRINT"{DOWN,3SPACE}VOLLSTAENDIG AUSDR
     UCKEN (J/N) ?":GOSUB 8200:AD$=A$           <153>
3025 OPEN 1,8,2,"{2SPACE}"+DN$:OPEN 2,8,15      <004>
3030 KO$=" 'F1' ENDE DER EINGABE{2SPACE}'F
     2' ABBRUCH{2SPACE}"                        <129>
3040 GOSUB 9500:GOSUB 8900:T=0:W=0              <042>
3100 FOR J=1 TO AE:I=0:FOR K=0 TO L(J)-1        <137>
3110 IF PEEK(1024+X(J)+40*Y(J)+K)<>32 THEN
      I=1:W=W-(J<>IN):K=L(J)-1                  <078>
3120 NEXT K:POKE 51725+J,I:NEXT J               <152>
3130 I=PEEK(51725+IN)                           <141>
3140 IF I=0 THEN AN=1:E=NE-1:GOTO 3300          <179>
3200 SYS 51205:IF PEEK(248)=0 THEN 3800         <128>
3210 AN=256*PEEK(250)+PEEK(249)                 <127>
3220 E=256*PEEK(252)+PEEK(251)                  <083>
3300 IF SU$="E"THEN 3330                        <191>
3310 IF AN>E AND GF=1 THEN 3750                 <029>
3320 IF AN>E THEN 3800                          <106>
3330 GOSUB 8800:GOSUB 9700:AZ=AN:IF W=0 TH
     EN 3500                                    <241>
3400 I=ID+6+3*AE+(IL+2)*(AN-1)                  <201>
3410 LO=PEEK(I):HI=PEEK(I+1):PRINT#2,"P"+C
     HR$(2)+CHR$(LO)+CHR$(HI)+CHR$(1)           <178>
3420 SYS 51761:AZ=AN                            <078>
3430 IF PEEK(248)=0 AND AN=E AND (SU$="E"
     OR GF=0) THEN 3800                         <103>
3440 IF PEEK(248)=0 AND AN=E THEN 3750          <074>
3450 IF PEEK(248)=0 THEN AN=AN+1:GOTO 3400      <008>
3500 GF=1:I=ID+6+3*AE+(IL+2)*(AN-1)             <023>
3510 LO=PEEK(I):HI=PEEK(I+1):PRINT#2,"P"+C
     HR$(2)+CHR$(LO)+CHR$(HI)+CHR$(1)           <022>
3520 SYS 49906:AN=AN+1                          <127>
3521 IF SU$="E"THEN 3530                        <190>
3523 KO$="{2SPACE}'D' DRUCKEN{2SPACE}'TAST
     E'{2SPACE}WEITERSUCHEN{2SPACE}":IF AD
     $="J"THEN 3525                             <058>
3524 GOSUB 9700:GOSUB 8100:IF A$<>"D"THEN
     3300                                       <076>
3525 GOSUB 9700:SYS 50843:IF PEEK(2)=0 THE
     N SYS 49376:GOTO 3300                      <180>
3526 CLOSE 1:CLOSE 2                            <046>
3527 OK$=KO$:KO$="{3SPACE}DRUCKER EINSCHAL
     TEN !{4SPACE}>TASTE<{3SPACE}":GOSUB 9
     700:KO$=OK$                                <199>
3528 SYS 50843:IF PEEK(2)<>0 THEN GOSUB 81
     00:GOTO 3528                               <237>
3529 GOSUB 9700:SYS 49376:OPEN 1,8,2,"{2SP
     ACE}"+DN$:OPEN 2,8,15:GOTO 3300            <050>
3530 KO$=" 'D'DRUCKEN 'A'AENDERN '+/-'BLAE
     TTERN ":GOSUB 9700                         <142>
3540 GOSUB 8100:IF A$<>"D"THEN 3550             <111>
3545 SYS 50843:IF PEEK(2)=0 THEN SYS 49376
     :GOTO 3530                                 <014>
3546 CLOSE 1:CLOSE 2:KO$=" DRUCKER EINSCHA
     LTEN ! 'F1' RUECKKEHR ":GOSUB 9700         <092>
3547 GOSUB 8100:IF A$="{F1}"THEN OPEN 1,8,
     2,"{2SPACE}"+DN$:OPEN 2,8,15:GOTO 353
     0                                          <107>
3548 SYS 50843:IF PEEK(2)<>0 THEN 3547          <143>
3549 SYS 49376:OPEN 1,8,2,"{2SPACE}"+DN$:O
     PEN 2,8,15:GOTO 3530                       <242>
3550 IF A$="+"AND AN<>NE THEN 3500              <232>
3551 IF A$="-"AND AN<>2 THEN AN=AN-2:GOTO
     3500                                       <169>
3552 IF A$="+"OR A$="-"THEN 3540                <224>
3554 IF A$<>"A"THEN AN=AZ:GOTO 3600             <032>
3555 KO$=" 'F1' ENDE DER EINGABE{2SPACE}'F
     2' ABBRUCH{2SPACE}"                        <146>
3560 IE=1:I=1:GOSUB 9700:GOSUB 9510:PRINT#
     2,"P"+CHR$(2)+CHR$(LO)+CHR$(HI)+CHR$(
     1)                                         <043>
3570 I=ID+6+3*AE:X=I+(NE-1)*(IL+2):Y=I+(AN
     -2)*(IL+2):FOR J=2 TO IL+1                 <233>
3575 POKE Y+J,PEEK(1022+40*Y(IN)+X(IN)+J):
     NEXT J:FOR J=0 TO IL+1                     <067>
3580 POKE X+J,PEEK(Y+J):NEXT J:FOR J=Y TO
     X:POKE J,PEEK(J+IL+2):NEXT J               <161>
3590 POKE 248,(NE-1)/256:POKE 247,NE-1-256
     *PEEK(248):POKE 189,IL+2:SYS 49514         <043>
3595 SYS 49979:GOTO 3530                        <096>
3600 AN=AN+1:IF AN<=E THEN KO$="'W' WEITER
     {2SPACE}'N' NEU SUCHEN{2SPACE}'F1' EN
     DE ":GOTO 3710                             <169>
3700 KO$="{6SPACE}'N' NEU SUCHEN{3SPACE}'F
     1' ENDE{6SPACE}"                           <031>
3710 GOSUB 9700:GOSUB 8100                      <128>
3720 IF A$="W"AND AN<=E THEN 3300               <046>
3730 IF A$="N"THEN 3030                         <036>
3740 IF A$<>"{F1}"THEN 3710                     <202>
3750 CLOSE 1:CLOSE 2:IF IE THEN 2710            <009>
3760 GOTO 800                                   <188>
3800 KO$="{2SPACE}DATENSATZ NICHT VORHANDE
     N{2SPACE}>TASTE<{2SPACE}"                  <046>
3810 GOSUB 9700:GOSUB 8100                      <228>
3820 IF SU$="E"THEN 3700                        <171>
3830 GOTO 3750                                  <246>
3960 :                                          <126>
3970 REM ******************************         <214>
3980 REM *      DRUCKERPARAMETER       *        <231>
3990 REM ******************************         <234>
4000 A$="DRUCKERPARAMETER":GOSUB 9000           <127>
4010 PRINT"{DOWN,3SPACE}BREITSCHRIFT (J/N)
      ?":GOSUB 8200:D$=A$:IF D$="J"THEN 40
     40                                         <131>
4020 PRINT"{DOWN,3SPACE}TABWERT (0-40) :{2
     SPACE,2LEFT}";                             <079>
4030 L=2:GOSUB 8300:POKE 4,ZA:IF ZA<0 OR Z
     A>40 THEN PRINT"{3UP}":GOTO 4020           <208>
4040 POKE 3,0:IF D$="J"THEN POKE 3,1            <074>
4050 GOTO 800                                   <224>
4960 :                                          <110>
4970 REM ******************************         <198>
4980 REM *      DISKOPERATIONEN        *        <158>
4990 REM ******************************         <218>
5000 SYS 52261                                  <095>
```

```
5010 GOSUB 8100:A=ASC(A$)-48                <078>
5020 IF A<1 OR A>6 THEN 5010                <192>
5030 GOSUB 8700:ON A GOTO 5100,5300,5400,5
     500,5600,800                           <234>
5100 A$="DIRECTORY":GOSUB 9000:OPEN 1,8,0,
     "$0"                                   <012>
5110 GET#1,A$,B$:GOSUB 8510:IF A THEN CLOS
     E 15:GOTO 5800                         <197>
5120 A$="DIRECTORY":GOSUB 9000:FOR I=1 TO
     16                                     <035>
5130 SYS 49753:IF ST THEN 5180              <164>
5140 GET A$:IF A$="{F2}"THEN 9600           <067>
5150 IF A$="{F1}"THEN CLOSE 15:GOTO 5800    <202>
5160 NEXT I:IF ST THEN 5180                 <079>
5170 PRINT:PRINT:GOSUB 8600:GOTO 5120       <234>
5180 PRINT"{RVSON}BLOCKS FREE{DOWN}":CLOSE
      15                                    <240>
5190 GOSUB 8600:GOTO 5800                   <078>
5300 A$="DATEI LOESCHEN":GOSUB 9000:F=44:L
     =14                                    <009>
5310 PRINT"{DOWN,3SPACE}NAME DER DATEI :";  <226>
5320 NA$="":GOSUB 8400                      <067>
5330 OPEN 1,8,15,"S:??"+NA$+","+NA$         <104>
5340 GOTO 5800                              <104>
5400 A$="DATEI UMBENENNEN":GOSUB 9000:PRIN
     T"{DOWN,3SPACE}ALTER NAME :";          <155>
5410 L=14:NA$=DN$:GOSUB 8400:MA$=NA$        <148>
5420 PRINT:PRINT"{3SPACE}NEUER NAME :";     <252>
5430 L=14:NA$="":GOSUB 8400:IF NA$=MA$ THE
     N 5000                                 <174>
5440 OPEN 1,8,15,"R:M-"+NA$+"=M-"+MA$       <152>
5450 PRINT#1,"R:I-"+NA$+"=I-"+MA$           <184>
5460 PRINT#1,"R:{2SPACE}"+NA$+"={2SPACE}"+
     MA$                                    <143>
5470 PRINT#1,"R:"+NA$+"="+MA$:CLOSE 1       <160>
5480 OPEN 1,8,15:INPUT#1,A,B$,C,D           <141>
5490 CLOSE 1:DN$=NA$:GOTO 5000              <180>
5500 A$="DISK FORMATIEREN":GOSUB 9000       <080>
5510 PRINT"{DOWN,3SPACE}DISKETTENNAME :";   <086>
5520 L=16:NA$="":GOSUB 8400                 <174>
5530 PRINT"{DOWN,3SPACE}DISKETTEN ID :";:M
     A$=NA$                                 <033>
5540 L=2:NA$="":GOSUB 8400                  <095>
5550 OPEN 1,8,15:PRINT#1,"M-W"CHR$(81)CHR$
     (0)CHR$(1)CHR$(255):CLOSE 1            <128>
5560 OPEN 1,8,15,"N:"+MA$+","+NA$           <183>
5570 PRINT;:GOTO 5800                       <248>
5600 OPEN 1,8,15,"V"                        <102>
5800 CLOSE 1:GOSUB 8500:GOTO 5000           <028>
5960 :                                      <094>
5970 REM ******************************     <182>
5980 REM *       PROGRAMM ENDE        *     <076>
5990 REM ******************************     <202>
6000 PRINT TAB(12)"{DOWN}SICHER (J/N) ?";   <206>
6010 GOSUB 8200:IF A$="N"THEN 800           <193>
6020 SYS 58236                              <133>
7960 :                                      <062>
7970 REM ******************************     <150>
7980 REM *  ALLGEMEINE UNTERROUTINEN  *     <004>
7990 REM ******************************     <170>
8000 POKE 211,X:POKE 214,Y:SYS 58640        <180>
8010 RETURN                                 <194>
8090 :                                      <192>
8100 POKE 198,0                             <134>
8110 GET A$:IF A$=""THEN 8110               <062>
8120 IF A$="{F2}"THEN 9600                  <033>
8130 RETURN                                 <058>
8190 :                                      <036>
8200 GOSUB 8100                             <149>
8210 IF A$<>"J"AND A$<>"N"THEN 8200         <031>
8220 RETURN                                 <057>
8290 :                                      <214>
8300 FOR I=1 TO L:POKE 256+I,0:NEXT I       <218>
8310 F=1:POKE 250,L:SYS 50877               <246>
8320 GOSUB 8420:ZA=VAL(NA$):RETURN          <254>
8390 :                                      <058>
8400 FOR I=1 TO L:POKE 256+I,ASC(MID$(NA$,
     I,1)+CHR$(0)):NEXT I                   <212>
8410 F=0:POKE 250,L:SYS 50885               <079>
8420 IF PEEK(250)=137 THEN 9600             <191>
8430 X=1024+PEEK(211)+40*PEEK(214)          <186>
8440 NA$="":FOR I=1 TO L:Y=PEEK(X+I)        <247>
8450 Y=Y-64*(Y<32 OR Y>95)-32*(Y>63 AND Y<
     96):NA$=NA$+CHR$(Y):NEXT I             <052>
8460 IF RIGHT$(NA$,1)=" "THEN NA$=LEFT$(NA
     $,LEN(NA$)-1):GOTO 8460                <219>
8470 IF NA$<>""OR F THEN PRINT:RETURN       <048>
8480 GOTO 8410                              <206>
8490 :                                      <084>
8500 GOSUB 8510:CLOSE 15:RETURN             <158>
8510 OPEN 15,8,15:INPUT#15,A,B$,C,D         <249>
8520 IF A=0 THEN RETURN                     <035>
8530 POKE 250,A:SYS 51927:A=PEEK(250)       <121>
8540 PRINT:IF A THEN PRINT"{2SPACE}"A;B$;C
     ;D                                     <136>
8550 CLOSE 15:A=1                           <039>
8590 :                                      <184>
8600 PRINT TAB(30)">RETURN<":IF ID THEN 81
     00                                     <086>
8610 GET A$:IF A$=""THEN 8610               <119>
8620 RETURN                                 <040>
8690 :                                      <028>
8700 X=(4+2*A)*40+1136                      <048>
8710 FOR I=13 TO 1 STEP -1                  <196>
8720 POKE X-27+I,PEEK(X-27+I) OR 128        <112>
8730 POKE X-I,PEEK(X-I) OR 128:NEXT I       <170>
8740 FOR I=0 TO 100:NEXT I:RETURN           <180>
8790 :                                      <130>
8800 POKE 49166,160:POKE 49170,4            <044>
8810 PRINT CHR$(14);:SYS 49152:RETURN       <219>
8890 :                                      <230>
8900 POKE 49166,4:POKE 49170,160            <241>
8910 GOTO 8810                              <174>
8990 :                                      <076>
9000 PRINT"{CLR,3DOWN}"TAB(20-LEN(A$)/2);A
     $:PRINT TAB(20-LEN(A$)/2);LEFT$(U$,LE
     N(A$))                                 <175>
9010 PRINT CHR$(142);:SYS 51171:RETURN      <164>
9090 :                                      <176>
9100 A=0:IF NA$=DN$ THEN 9200               <115>
9110 SYS(57812)"M-"+NA$,8:POKE 780,0:POKE
     781,0:POKE 782,160:SYS 65493           <152>
9120 GOSUB 8500:IF A THEN RETURN            <126>
9130 SYS(57812)"I-"+NA$,8:POKE 780,0:POKE
     781,152:POKE 782,58:SYS 65493          <252>
9140 GOSUB 8500:IF A THEN RETURN            <146>
9150 DN$=NA$                                <118>
9190 :                                      <020>
9200 SYS 49334                              <048>
9210 AE=PEEK(ID):IN=PEEK(ID+1)              <252>
9220 IF AE>30 THEN RETURN                   <182>
9240 NE=PEEK(ID+4)+256*PEEK(ID+5)           <147>
9250 FOR I=1 TO AE:X=ID+3+3*I:L(I)=PEEK(X)
     :X(I)=PEEK(X+1):Y(I)=PEEK(X+2):NEXT I  <064>
9260 RL=0:FOR I=1 TO AE:RL=RL+L(I):NEXT     <140>
9270 IL=L(IN):RETURN                        <099>
9290 :                                      <122>
9300 PRINT"{DOWN,3SPACE}NAME DER DATEI :";
     :L=14                                  <060>
9310 F=42:NA$=DN$:GOSUB 8400:RETURN         <024>
9490 :                                      <068>
9500 I=1:GOSUB 8800:GOSUB 9700              <054>
9510 X=X(I):Y=Y(I):GOSUB 8000               <169>
9520 X=1024+40*Y+X                          <102>
9530 POKE 252,X/256:POKE 251,X-256*PEEK(25
     2):POKE 253,L(I):POKE 254,I            <023>
9540 POKE 250,0:SYS 50625:I=PEEK(254)       <060>
9550 IF PEEK(250)=133 THEN RETURN           <190>
9560 IF PEEK(250)=137 THEN 9600             <094>
9570 IF I<1 THEN I=AE                       <091>
9580 IF I>AE THEN I=1                       <007>
9590 GOTO 9510                              <052>
9599 :                                      <177>
9600 CLOSE 1:CLOSE 2:CLOSE 15               <011>
9610 SYS 50821:IF IE=3 THEN NE=NE+1:GOTO 2
     500                                    <114>
9620 IF IE=2 THEN 2500                      <060>
9630 IF IE=1 THEN 2710                      <014>
9640 GOTO 800                               <226>
9690 :                                      <012>
9700 X=1:Y=24:GOSUB 8000:PRINT"{RVSON}";KO
     $;"{RVOFF}";:POKE 2023,32:RETURN       <057>
9790 :                                      <114>
9800 PRINT#15,"I":SYS 52182:FB=PEEK(250)+2
     56*PEEK(251)                           <068>
9810 INPUT#15,A,B$,C,D:RETURN               <234>
9890 :                                      <214>
9900 OPEN 15,8,15:GOSUB 9800:FB=INT(252*FB
     /RL):FB=INT(FB/(1+IL/254)):CLOSE 15    <103>
9910 KO$=" 0000 FREIE, 0000 BELEGTE DATENS
```

```
     RETZE ":IF FB>9999 THEN FB=9999            <179>
9920 X=NE:IF NE>9999 THEN X=9999                <071>
9930 A$=RIGHT$(STR$(FB),LEN(STR$(FB))-1):B
     $=RIGHT$(STR$(X-1),LEN(STR$(X-1))-1)       <065>
9940 KO$=LEFT$(KO$,5-LEN(A$))+A$+MID$(KO$,
     6,12-LEN(B$))+B$+RIGHT$(KO$,21)            <046>
9950 GOSUB 9700:GOSUB 8100:RETURN               <250>
9960 :                                          <028>
9970 REM ****************************           <116>
9980 REM *            DATAS             *       <159>
9990 REM ****************************           <138>
10000 DATA 000,000,000,015,255,240,031,255
      ,248,056,000,028,049,129,140,051          <076>
10010 DATA 195,204,049,129,140,048,000,012
      ,051,000,204,049,000,140,048,255          <176>
10020 DATA 012,056,000,012,031,255,248,015
      ,255,240,000,000,000,015,255,240          <111>
10030 DATA 022,102,104,043,102,212,086,102
      .106.127,255,254,000,000,000              <125>
```

Listing 1. »Polydat« (Schluß)

```
Name : d2                    c000 cee0
-----------------------------------
c000 : 78 a5 01 29 fe 85 01 a0   12
c008 : 00 84 fb 84 fd a2 a0 86   5e
c010 : fc a2 04 86 fe a2 18 a0   d6
c018 : 27 b1 fb 91 fd 88 10 f9   a1
c020 : 18 a5 fb 69 28 85 fb 85   e0
c028 : fd 90 04 e6 fc e6 fe ca   e4
c030 : d0 e5 a5 01 09 01 85 01   2d
c038 : 58 60 78 a5 01 29 fe 85   f3
c040 : 01 a2 00 86 fd a9 3a 85   84
c048 : fe 8a a8 91 fd c8 d0 fb   49
c050 : e6 fe a5 fe c9 a0 d0 f3   cb
c058 : a9 9d 85 f8 a9 3a 85 f9   c7
c060 : a9 a0 85 fe a0 00 b1 fd   67
c068 : d0 2f ee 98 3a a9 00 85   9a
c070 : fa 84 fb 86 fc e6 fa c8   00
c078 : c0 28 f0 04 b1 fd f0 f5   c4
c080 : 98 48 a0 03 b9 f9 00 91   53
c088 : f8 88 d0 f8 18 a5 f8 69   7d
c090 : 03 85 f8 d0 02 e6 f9 68   be
c098 : a8 c8 c0 28 90 c8 18 a5   d4
c0a0 : fd 69 28 85 fd 90 02 e6   47
c0a8 : fe e8 e0 18 d0 b6 a5 01   b1
c0b0 : 09 01 85 01 58 60 78 a5   71
c0b8 : 01 29 fe 85 01 a0 00 84   dc
c0c0 : fd a9 a0 85 fe b1 fd d0   82
c0c8 : 04 a9 20 91 fd c8 d0 f5   30
c0d0 : e6 fe a5 fe c9 a4 d0 ed   5f
c0d8 : a5 01 09 01 85 01 58 60   e3
c0e0 : a9 04 a2 04 a0 07 20 ba   ed
c0e8 : ff 20 c0 ff a2 04 20 c9   86
c0f0 : ff a9 00 85 fa a9 04 85   8d
c0f8 : fb a2 18 a9 0d 20 d2 ff   9d
c100 : a0 27 b1 fa c9 20 d0 05   ea
c108 : 88 10 f7 30 41 a5 03 f0   cb
c110 : 07 a9 0e 20 d2 ff d0 0c   fc
c118 : a4 04 f0 08 a9 20 20 d2   bd
c120 : ff 88 d0 f8 a0 00 b1 fa   7d
c128 : c9 20 b0 06 18 69 40 4c   55
c130 : 46 c1 c9 40 b0 03 4c 46   b2
c138 : c1 c9 60 b0 06 18 69 80   d4
c140 : 4c 46 c1 18 69 40 20 d2   e1
c148 : ff c8 c0 28 d0 d8 18 a5   60
c150 : fa 69 28 85 fa 90 02 e6   c4
c158 : fb ca d0 9f a9 0d 20 d2   09
c160 : ff 20 cc ff a9 04 20 c3   65
c168 : ff 60 a5 f8 d0 09 a5 f7   fc
c170 : c9 01 d0 03 4c 41 c2 18   58
c178 : a9 9e 6d 98 3a 6d 98 3a   c5
c180 : 6d 98 3a 85 fb 85 fd a9   b0
c188 : 3a 85 fc 85 fe a9 01 85   c1
c190 : f9 a9 00 85 fa e6 f9 d0   7f
c198 : 02 e6 fa a5 fa c5 f8 90   63
c1a0 : 08 a5 f9 c5 f7 90 02 b0   1f
c1a8 : 0d 18 a5 fd 65 bd 85 fd   41
c1b0 : 90 02 e6 fe d0 df 18 a5   92
c1b8 : fd 65 bd 85 f9 a5 fe 69   23
c1c0 : 00 85 fa a0 02 b1 fb d1   97
c1c8 : f9 90 07 d0 23 c8 c4 bd   ec
c1d0 : d0 f3 18 a5 fb 65 bd 85   41
c1d8 : fb 90 02 e6 fc a5 fc c5   f5
c1e0 : fe 90 e0 d0 08 a5 fb c5   a2
c1e8 : fd 90 d8 f0 d6 4c 41 c2   dc
c1f0 : a0 00 b1 f9 99 00 01 c8   6b
c1f8 : c4 bd d0 f6 18 a5 f9 65   0f
c200 : bd 85 fd a5 fa 69 00 85   ba
c208 : fe 90 06 a0 00 b1 f9 91   7d
c210 : fd 38 a5 fd e9 01 85 fd   0b
c218 : b0 02 c6 fe 38 a5 f9 e9   c7
c220 : 01 85 f9 b0 02 c6 fa a5   06
c228 : fc c5 fa 90 de d0 08 a5   b7
c230 : fb c5 f9 90 d6 f0 d4 b9   5a
c238 : 00 01 91 fb c8 c4 bd d0   e8
c240 : f6 e6 f7 d0 02 e6 f8 a5   48
c248 : f8 cd 9d 3a b0 03 4c 77   19
c250 : c1 a5 f7 cd 9c 3a 90 f6   67
c258 : 60 a2 01 20 c6 ff 20 e4   04
c260 : ff 20 e4 ff 20 e4 ff 85   dd
c268 : 63 20 e4 ff 85 62 a2 90   2b
c270 : 38 20 49 bc a9 0d 20 d2   cb
c278 : ff a9 20 a2 05 20 d2 ff   45
c280 : ca d0 fa 20 dd bd a0 01   c5
c288 : b9 00 01 f0 06 20 d2 ff   4c
c290 : c8 d0 f5 a9 0a 38 e5 d3   14
c298 : aa a9 20 20 d2 ff ca d0   1d
c2a0 : fa 20 e4 ff a4 90 d0 46   82
c2a8 : c9 22 d0 f5 20 e4 ff c9   32
c2b0 : 22 f0 06 20 d2 ff 4c ac   88
c2b8 : c2 a9 1e 38 e5 d3 aa a9   d8
c2c0 : 20 20 d2 ff ca d0 fa 20   04
c2c8 : e4 ff c9 20 f0 f9 a0 00   84
c2d0 : 20 d2 ff 20 e4 ff f0 0a   83
c2d8 : c8 c0 05 d0 f3 20 e4 ff   2f
c2e0 : d0 fb c0 05 b0 08 c8 a9   40
c2e8 : 20 20 d2 ff d0 f4 20 cc   9c
c2f0 : ff 60 a2 01 20 c6 ff a9   74
c2f8 : 00 85 fe 18 a5 fe 65 fe   63
c300 : 65 fe aa 18 a9 00 7d 9f   62
c308 : 3a 85 fb a9 04 85 fc a0   da
c310 : 28 18 a5 fb 7d a0 3a 85   fe
c318 : fb 90 02 e6 fc 88 d0 f1   f4
c320 : a0 00 20 e4 ff 91 fb c8   73
c328 : 98 dd 9e 3a d0 f4 e6 fe   ec
c330 : a5 fe cd 98 3a d0 c4 20   58
c338 : cc ff 60 a2 01 20 c9 ff   a8
c340 : a9 00 85 fe 18 a5 fe 65   a0
c348 : fe 65 fe aa 18 a9 00 7d   d8
c350 : 9f 3a 85 fb a9 04 85 fc   b8
c358 : a0 28 18 a5 fb 7d a0 3a   6a
c360 : 85 fb 90 02 e6 fc 88 d0   61
c368 : f1 a0 00 b1 fb 20 d2 ff   ec
c370 : c8 98 dd 9e 3a d0 f4 e6   9b
c378 : fe a5 fe cd 98 3a d0 c4   ea
c380 : 20 cc ff 60 a9 0e 20 d2   43
c388 : ff a9 80 8d 8a 02 a9 00   8d
c390 : 85 fe 85 fd a6 fe a4 fd   a6
c398 : e8 f0 04 ca 4c a2 c3 a2   81
c3a0 : 27 88 e0 28 90 03 a2 00   f4
c3a8 : c8 c8 f0 01 88 c0 18 90   41
c3b0 : 02 a0 17 84 fd 86 fe a0   aa
c3b8 : ff 18 8a 69 d8 85 fb a9   90
c3c0 : 03 85 fc 18 a5 fb 69 28   f8
c3c8 : 85 fb 90 02 e6 fc c8 c4   b2
c3d0 : fd d0 f0 20 ab c5 20 e4   a9
c3d8 : ff f0 fb c9 11 d0 07 20   7b
c3e0 : ab c5 e6 fd d0 ae c9 1d   cb
c3e8 : d0 07 20 ab c5 e6 fe d0   ea
c3f0 : a3 c9 91 d0 08 20 ab c5   b2
c3f8 : c6 fd 4c 94 c3 c9 9d d0   05
c400 : 08 20 ab c5 c6 fe 4c 94   7b
c408 : c3 c9 14 d0 29 20 ab c5   9d
c410 : a5 fe f0 80 38 a5 fb e5   ed
c418 : fe 85 fb b0 02 c6 fc a4   81
c420 : fe b1 fb 88 91 fb c8 c8   b4
c428 : c0 28 d0 f5 88 a9 20 91   69
c430 : fb c6 fe 4c 94 c3 c9 94   8f
c438 : d0 2f 20 ab c5 38 a5 fb   ca
c440 : e5 fe 85 fb b0 02 c6 fc   b5
c448 : a0 27 b1 fb c9 20 f0 07   d7
c450 : c9 60 f0 03 4c 94 c3 88   6f
c458 : b1 fb c8 91 fb 88 c4 fe   80
c460 : d0 f5 a9 20 91 fb 4c 94   ed
c468 : c3 c9 13 d0 06 20 ab c5   8a
c470 : 4c 84 c3 c9 93 d0 06 85   0b
c478 : fa 20 ab c5 60 c9 8d d0   52
c480 : 0c 20 ab c5 a9 00 85 fe   ef
c488 : e6 fd 4c 94 c3 c9 0d f0   b3
c490 : f0 c9 85 d0 0b 85 fa 20   e9
c498 : ab c5 a9 00 8d 8a 02 60   86
c4a0 : c9 89 f0 f1 c9 86 d0 65   87
c4a8 : 20 ab c5 a5 fd f0 5b a9   ec
c4b0 : 00 85 fb a9 04 85 fc a4   51
c4b8 : fd 18 a5 fb 69 28 85 fb   90
c4c0 : 90 02 e6 fc 88 d0 f2 38   f6
c4c8 : a5 fb e9 28 85 f9 a5 fc   a3
c4d0 : e9 00 85 fa a0 27 b1 fb   7c
c4d8 : 91 f9 88 10 f9 18 a5 f9   75
c4e0 : 69 28 85 f9 90 02 e6 fa   a8
c4e8 : 18 a5 fb 69 28 85 fb 90   bf
c4f0 : 02 e6 fc a5 fc c9 07 d0   35
c4f8 : db a5 fb c9 c0 d0 d5 a9   1b
c500 : 20 a0 27 91 f9 88 10 fb   88
c508 : c6 fd 4c 94 c3 c9 8a d0   c9
c510 : 60 20 ab c5 a9 00 85 fb   cc
c518 : a9 04 85 fc a0 16 18 a5   2b
c520 : fb 69 28 85 fb 90 02 e6   a5
c528 : fc 88 d0 f2 18 a5 fb 69   6c
c530 : 28 85 f9 a5 fc 69 00 85   74
c538 : fa a5 fd c9 17 f0 26 a2   94
c540 : 17 a0 27 b1 fb 91 f9 88   ed
c548 : 10 f9 38 a5 f9 e9 28 85   b2
c550 : f9 b0 02 c6 fa 38 a5 fb   fb
c558 : e9 28 85 fb b0 02 c6 fc   66
c560 : ca e4 fd d0 dc a9 20 a0   13
c568 : 27 91 f9 88 10 fb 4c 94   22
c570 : c3 c9 20 b0 03 4c d6 c3   ab
c578 : c9 ea b0 f9 20 88 c5 a0   c0
c580 : 00 91 fb e6 fe 4c 94 c3   51
c588 : c9 60 90 04 c9 a0 90 e5   d6
c590 : c9 40 90 04 c9 60 90 08   10
c598 : c9 a0 90 07 c9 c0 b0 03   22
c5a0 : 38 e9 40 c9 c0 90 03 38   23
c5a8 : e9 80 60 a0 00 b1 fb 29   cd
c5b0 : 80 f0 07 b1 fb 29 7f 91   ca
c5b8 : fb 60 b1 fb 09 80 91 fb   a2
c5c0 : 60 a0 00 20 ad c5 84 f9   83
c5c8 : 20 e4 ff f0 fb a4 f9 48   d6
c5d0 : 20 ad c5 68 c9 1d d0 0b   24
c5d8 : c8 c4 fd 08 88 28 f0 e3   d8
c5e0 : c8 d0 e0 c9 9d d0 07 c0   80
c5e8 : 01 90 d8 88 10 d5 c9 85   5a
c5f0 : d0 03 85 fa 60 c9 89 f0   5f
c5f8 : f9 c9 11 d0 03 e6 fe 60   58
c600 : c9 0d f0 f9 c9 91 d0 03   3d
c608 : c6 fe 60 c9 8d f0 f9 c9   7a
c610 : 14 d0 1c c0 01 90 ac 88   04
c618 : 84 f9 c8 b1 fb 88 91 fb   43
c620 : c8 c8 c4 fd d0 f5 88 a9   6f
c628 : 20 91 fb a4 f9 10 94 c9   aa
c630 : 94 d0 23 84 f9 a4 fd 88   53
c638 : b1 fb c9 20 f0 04 c9 60   74
c640 : d0 0f 88 b1 fb c8 91 fb   34
c648 : 88 c4 f9 d0 f5 a9 20 91   1b
c650 : fb a4 f9 4c c3 c5 c9 93   5e
c658 : d0 0d a4 fd 88 a9 20 91   11
c660 : fb 88 10 fb c8 10 ec c9   77
c668 : 20 90 e8 c4 fd f0 e4 c9   31
c670 : 80 90 04 c9 a0 90 dc 20   b5
c678 : 88 c5 91 fb c8 c4 fd d0   13
c680 : 01 88 4c c3 c5 a2 04 68   a3
c688 : 95 f9 ca 10 fa a2 ff 9a   c8
c690 : a2 00 b5 f9 48 e8 e0 05   38
c698 : d0 f8 60 a9 01 a2 bc a0   8b
c6a0 : c6 20 bd ff a9 04 a2 04   33
c6a8 : a0 07 20 ba ff 20 c0 ff   2f
c6b0 : b0 02 a9 00 85 02 a9 04   e3
c6b8 : 20 c3 ff 60 20 a9 30 85   e1
c6c0 : fd a9 3a d0 06 a9 20 85   74
c6c8 : fd a9 ff 85 fe a9 00 85   93
c6d0 : fb a9 04 85 fc a4 d6 18   d2
c6d8 : a5 fb 69 28 85 fb 90 02   59
c6e0 : e6 fc 88 d0 f2 18 a5 fb   ff
c6e8 : 65 d3 85 fb 90 02 e6 fc   c6
c6f0 : a0 00 a9 3e 91 fb c8 b9   52
c6f8 : 00 01 d0 02 a9 20 20 88   1a
c700 : c5 91 fb c8 c4 fa f0 ef   6d
c708 : 90 ed a9 3c 91 fb a0 01   fe
c710 : 20 ad c5 84 f9 20 e4 ff   3d
c718 : f0 fb 48 a4 f9 20 ad c5   8f
c720 : 68 c9 0d d0 03 85 fa 60   d3
c728 : c9 85 f0 f9 c9 89 f0 f5   c8
c730 : c9 13 f0 da c9 1d d0 07   f1
c738 : c4 fa b0 d4 c8 d0 d1 c9   2e
c740 : 9d d0 07 c0 01 f0 c9 88   ef
c748 : 10 c6 c9 93 d0 0b a4 fa   8e
c750 : a9 20 91 fb 88 d0 fb f0   ce
c758 : b5 c9 14 d0 1d c0 01 f0   cf
c760 : af 84 f9 b1 fb 88 91 fb   48
c768 : c8 c8 c4 fa f0 f5 90 f3   0e
```

Listing 2. Der Maschinensprache-Teil (bitte mit dem MSE eingeben) von »Polydat« wird automatisch nachgeladen

```
c770 : 88 a9 20 91 fb a4 f9 88  e5
c778 : d0 96 c9 94 d0 23 84 f9  c4
c780 : a4 fa b1 fb a4 f9 c9 20  0f
c788 : f0 04 c9 60 d0 82 a4 fa  a2
c790 : 88 b1 fb c8 91 fb 88 c4  ad
c798 : f9 d0 f5 a9 20 91 fb d0  cc
c7a0 : 08 c9 80 90 07 c9 a0 b0  62
c7a8 : 03 4c 10 c7 c9 22 f0 f9  34
c7b0 : c9 2a f0 f5 c9 2c f0 f1  2f
c7b8 : c9 3a f0 ed c9 3f f0 e9  c6
c7c0 : c9 3d f0 e5 c5 fd b0 04  38
c7c8 : c9 20 d0 dd c5 fe 90 04  30
c7d0 : c9 a0 d0 d5 20 88 c5 91  59
c7d8 : fb c4 fa f0 cc c8 d0 c9  fc
c7e0 : c8 d0 e1 a9 00 85 fb a9  2d
c7e8 : d8 85 fc a2 18 a0 27 a9  8d
c7f0 : 07 91 fb 88 10 fb 18 a5  5c
c7f8 : fb 69 28 85 fb 90 02 e6  7d
c800 : fc ca d0 e9 60 ad 9c 3a  2d
c808 : 85 fb ad 9d 3a 85 fc a9  c1
c810 : 00 85 60 85 61 85 fa a9  1d
c818 : d8 85 fd a9 01 85 f9 a9  df
c820 : 03 85 fe ad 99 3a 0a 18  1f
c828 : 6d 99 3a a8 a5 fd 79 9c  6f
c830 : 3a 85 fd 90 02 e6 fe b9  85
c838 : 9b 3a 85 bd 18 69 02 85  e9
c840 : b6 b9 9d 3a a8 18 a5 fd  5f
c848 : 69 28 85 fd 90 02 e6 fe  99
c850 : 88 10 f2 38 a5 fb e5 f9  6a
c858 : 85 f7 a5 fc e5 fa 85 f8  20
c860 : 18 a5 f7 69 02 85 f7 90  c3
c868 : 02 e6 f8 46 f8 66 f7 18  b7
c870 : a5 f7 65 f9 85 f7 a5 f8  4a
c878 : 65 fa 85 f8 38 a5 f7 e9  3f
c880 : 01 85 f7 b0 02 c6 f8 a5  dd
c888 : f8 c5 61 d0 0b a5 f7 c5  1e
c890 : 60 d0 05 a9 00 85 f8 60  a0
c898 : a5 f7 85 60 a5 f8 85 61  a1
c8a0 : a0 00 a9 a0 85 05 a9 3a  5a
c8a8 : 85 06 a2 03 18 a5 05 6d  d7
c8b0 : 98 3a 85 05 ca d0 f5 98  a4
c8b8 : 48 a0 01 a2 00 e4 61 d0  33
c8c0 : 04 c4 60 f0 17 18 a5 05  2f
c8c8 : 65 b6 85 05 90 02 e6 06  4b
c8d0 : c8 d0 01 e8 e4 61 d0 ed  d6
c8d8 : c4 60 d0 e9 68 a8 b1 fd  cc
c8e0 : c9 2a f0 25 c9 3f f0 21  3c
c8e8 : d1 05 b0 0b a5 60 85 fb  35
c8f0 : a5 61 85 fc 4c 53 c8 f0  ab
c8f8 : 0b a5 60 85 f9 a5 61 85  fc
c900 : fa 4c 53 c8 c8 c4 bd d0  59
c908 : d5 a5 60 85 f9 a5 61 85  d6
c910 : fa a0 00 a9 a0 85 05 a9  2d
c918 : 3a 85 06 a2 03 18 a5 05  7c
c920 : 6d 98 3a 85 05 ca d0 f5  ee
c928 : 98 48 38 a5 05 e5 b6 b0  63
c930 : 02 c6 06 38 e5 b6 b0 02  f9
c938 : c6 06 85 05 a2 00 a0 00  b0
c940 : 18 a5 05 65 b6 85 05 90  e6
c948 : 02 e6 06 c8 d0 01 e8 e4  da
c950 : fa d0 ed c4 f9 d0 e9 68  65
c958 : a8 b1 fd c9 2a f0 1f c9  cc
c960 : 3f f0 1b b1 05 d1 fd b0  4d
c968 : 10 38 a5 f9 e9 01 85 fb  f2
c970 : a5 fa e9 00 85 fc 4c 96  ab
c978 : c9 c8 c4 bd d0 db a5 fa  07
c980 : d0 06 a5 f9 c9 01 f0 e2  2a
c988 : 38 a5 f9 e9 01 85 f9 b0  d4
c990 : 02 c6 fa 4c 11 c9 a0 00  1f
c998 : a9 a0 85 05 a9 3a 85 06  22
c9a0 : a2 03 18 a5 05 6d 98 3a  11
c9a8 : 85 05 ca d0 f5 98 48 a2  07
c9b0 : 00 a0 00 18 a5 05 65 b6  89
c9b8 : 85 05 90 02 e6 06 c8 d0  88
c9c0 : 01 e8 e4 fc d0 ed c4 fb  95
c9c8 : d0 e9 68 a8 b1 fd c9 2a  42
c9d0 : f0 04 c9 3f d0 06 a4 db  a4
c9d8 : 88 4c ea c9 d1 05 b0 05  8c
c9e0 : a9 01 85 f8 60 c8 c4 bd  65
c9e8 : d0 e2 e6 fb d0 02 e6 fc  15
c9f0 : a5 fc cd 9d 3a 90 14 a5  fe
c9f8 : fb cd 9c 3a 90 0d a5 fb  48
ca00 : e9 01 85 fb b0 02 c6 fc  7b
ca08 : 4c e0 c9 4c 96 c9 00 00  78
ca10 : 00 00 00 00 00 00 00 00  11
ca18 : 00 00 00 00 00 00 00 00  19
ca20 : 00 00 00 00 00 00 00 00  21
ca28 : 00 00 00 00 00 00 00 00  29
ca30 : 00 a2 01 20 c6 ff a0 01  b7
ca38 : 84 06 98 0a 18 65 06 a8  3d
ca40 : b9 9b 3a 85 f9 b9 9c 3a  5a
ca48 : 85 fa b9 9d 3a 85 fd a4  7d
ca50 : 06 b9 0d ca d0 11 a0 00  e8
ca58 : 98 48 20 e4 ff 68 a8 c8  30
ca60 : c4 f9 d0 f4 4c c4 ca a9  5d
ca68 : 00 85 fb a9 04 85 fc 18  ef
ca70 : a5 fb 65 fa 85 fb 90 02  4a
ca78 : e6 fc a4 fd f0 0e 18 a5  f0
ca80 : fb 69 28 85 fb 90 02 e6  05
ca88 : fc 88 d0 f2 b1 fb c9 2a  d1
ca90 : d0 0e 98 48 20 e4 ff 68  90
ca98 : a8 c8 c4 f9 d0 f4 f0 24  d5
caa0 : c9 3f d0 06 20 e4 ff 4c  bf
caa8 : bf ca 98 48 20 e4 ff aa  7a
cab0 : 68 a8 8a d1 fb f0 08 20  f1
cab8 : cc ff a9 00 85 f8 60 c8  21
cac0 : c4 f9 d0 c8 a4 06 cc 98  ad
cac8 : 3a f0 04 c8 4c 38 ca 20  86
cad0 : cc ff a9 01 85 f8 60 a5  13
cad8 : fa a2 00 86 fa c9 1a d0  fc
cae0 : 2f a0 23 b9 ec ca 20 d2  ab
cae8 : ff 88 d0 f7 60 21 20 4e  8b
caf0 : 45 4e 52 45 46 54 4e 45  64
caf8 : 20 5a 54 55 48 43 53 42  76
cb00 : 49 45 52 48 43 53 20 45  63
cb08 : 54 54 49 42 20 20 20 11  c7
cb10 : c9 3e d0 27 a0 1b b9 1f  19
cb18 : cb 20 d2 ff 88 d0 f7 60  57
cb20 : 21 20 4e 45 44 4e 41 48  da
cb28 : 52 4f 56 20 54 48 43 49  e3
cb30 : 4e 20 49 45 54 41 44 20  2a
cb38 : 20 20 11 c9 3f d0 29 a0  46
cb40 : 1d b9 4a cb 20 d2 ff 88  ef
cb48 : d0 f7 60 21 20 4e 45 44  62
cb50 : 4e 41 48 52 4f 56 20 53  6a
cb58 : 54 49 45 52 45 42 20 49  66
cb60 : 45 54 41 44 20 20 20 11  4e
cb68 : c9 48 d0 29 a0 1d b9 77  77
cb70 : cb 20 d2 ff 88 d0 f7 60  af
cb78 : 21 20 45 54 54 45 4b 53  c8
cb80 : 49 44 20 46 55 41 20 5a  51
cb88 : 54 41 4c 50 20 4e 49 45  be
cb90 : 4b 20 20 20 11 c9 4a d0  22
cb98 : 39 a0 1d b9 a4 cb 20 d2  6f
cba0 : ff 88 d0 f7 60 21 20 4e  43
cba8 : 45 47 45 4c 4e 49 45 20  f0
cbb0 : 45 54 54 45 4b 53 49 44  da
cbb8 : 20 45 54 54 49 42 20 20  82
cbc0 : 20 11 a2 ff e2 ed 00 ff  af
cbc8 : a2 ff ef 11 ae 01 bb 46  f6
cbd0 : 00 11 85 fa 60 00 a2 0f  c8
cbd8 : 20 c9 ff a0 04 b9 18 cc  f9
cbe0 : 20 d2 ff 88 10 f7 20 cc  55
cbe8 : ff a2 0f 20 c6 ff 20 e4  b7
cbf0 : ff 85 fa a2 0f 20 c9 ff  de
cbf8 : a0 04 b9 1d cc 20 d2 ff  c5
cc00 : 88 10 f7 20 cc ff a2 0f  08
cc08 : 20 c6 ff 20 e4 ff 85 fb  ec
cc10 : 20 e4 ff 4c cc ff 00 00  f8
cc18 : 02 fa 52 2d 4d 02 fc 52  4f
cc20 : 2d 4d a9 00 2c a9 01 85  7d
cc28 : fc a0 00 a5 fc d0 0d b9  27
cc30 : f6 cc 20 d2 ff c8 c0 44  c0
cc38 : d0 f5 f0 0b b9 83 cd 20  cf
cc40 : d2 ff c8 c0 44 d0 f5 a0  40
cc48 : 00 b9 3a cd 20 d2 ff c8  97
cc50 : c0 49 d0 f5 a2 01 a9 20  c1
cc58 : 20 d2 ff a9 7d 20 d2 ff  3a
cc60 : a0 24 a9 20 20 d2 ff 88  2a
cc68 : d0 fa a0 00 b9 c7 cd 20  2f
cc70 : d2 ff c8 c0 0b d0 f5 8a  b0
cc78 : 18 69 30 20 d2 ff a0 00  05
cc80 : b9 d1 cd 20 d2 ff c8 c0  6b
cc88 : 05 d0 f5 8a 48 0a 0a 0a  d5
cc90 : 0a a8 18 69 10 85 fb a6  8c
cc98 : fc f0 06 b9 4d ce 4c a4  8b
cca0 : cc b9 ed cd 20 d2 ff c8  a8
cca8 : c4 fb d0 eb a0 07 a9 20  44
ccb0 : 20 d2 ff 88 d0 f8 a9 7d  c1
ccb8 : 20 d2 ff a9 0d 20 d2 ff  93
ccc0 : 68 aa e8 e0 07 d0 8f a9  5c
ccc8 : 20 20 d2 ff a9 7d 20 d2  59
ccd0 : ff a0 24 a9 20 20 d2 ff  ac
ccd8 : 88 d0 f8 a9 7d 20 d2 ff  60
cce0 : a9 0d 20 d2 ff a0 00 b9  eb
cce8 : d5 cd 20 d2 ff c8 c0 28  a0
ccf0 : d0 f5 60 4c df c7 8e 93  fa
ccf8 : 11 11 20 20 20 20 20 20  62
cd00 : 20 20 d5 c0 c0 c0 c0 c0  54
cd08 : c0 c0 c0 c0 c0 c0 c0 c0  07
cd10 : c0 c0 c0 c0 c0 c0 c0 c0  0f
cd18 : c9 0d 20 20 20 20 20 20  38
cd20 : 20 20 dd 20 48 20 41 20  97
cd28 : 55 20 50 20 54 20 4d 20  61
cd30 : 45 20 4e 20 55 20 45 20  c9
cd38 : dd 0d 20 20 20 20 20 20  6c
cd40 : 20 20 ca c0 c0 c0 c0 c0  d2
cd48 : c0 c0 c0 c0 c0 c0 c0 c0  47
cd50 : c0 c0 c0 c0 c0 c0 c0 c0  4f
cd58 : cb 11 0d 20 d5 c0 c0 c0  db
cd60 : c0 c0 c0 c0 c0 c0 c0 c0  5f
cd68 : c0 c0 c0 c0 c0 c0 c0 c0  67
cd70 : c0 c0 c0 c0 c0 c0 c0 c0  6f
cd78 : c0 c0 c0 c0 c0 c0 c0 c0  77
cd80 : c0 c9 0d 8e 93 11 11 20  80
cd88 : 20 20 20 20 20 20 20 d5  f4
cd90 : c0 c0 c0 c0 c0 c0 c0 c0  8f
cd98 : c0 c0 c0 c0 c0 c0 c0 c0  97
cda0 : c0 c0 c0 c0 c0 c9 0d 20  d7
cda8 : 20 20 20 20 20 20 20 dd  24
cdb0 : 20 20 20 44 49 53 4b 4f  6c
cdb8 : 50 45 52 41 54 49 4f 4e  d1
cdc0 : 45 4e 20 20 20 dd 0d dd  19
cdc8 : 20 20 dd 20 20 20 20 20  38
cdd0 : 28 20 29 20 20 20 ca c0  06
cdd8 : c0 c0 c0 c0 c0 c0 c0 c0  d7
cde0 : c0 c0 c0 c0 c0 c0 c0 c0  df
cde8 : c0 c0 c0 c0 c0 c0 c0 c0  e7
cdf0 : c0 c0 c0 c0 c0 c0 c0 c0  ef
cdf8 : c0 c0 c0 cb 0d 44 41 54  62
ce00 : 45 49 20 45 49 4e 52 49  7d
ce08 : 43 48 54 45 4e 44 41 54  e2
ce10 : 45 4e 20 45 49 4e 47 45  dc
ce18 : 42 45 4e 20 20 44 41 54  66
ce20 : 45 4e 20 53 55 43 48 45  1a
ce28 : 4e 20 20 20 20 44 52 55  ab
ce30 : 43 4b 45 52 50 41 52 41  8f
ce38 : 4d 45 54 45 52 44 49 53  f9
ce40 : 4b 4f 50 45 52 41 54 49  03
ce48 : 4f 4e 45 4e 20 50 52 4f  46
ce50 : 47 52 41 4d 4d 20 45 4e  42
ce58 : 44 45 20 20 20 44 49 52  39
ce60 : 45 43 54 4f 52 59 20 20  f7
ce68 : 20 20 20 20 20 44 41 54  76
ce70 : 45 49 20 4c 4f 45 53 43  de
ce78 : 48 45 4e 20 20 44 41 54  cc
ce80 : 45 49 20 55 4d 42 45 4e  b5
ce88 : 45 4e 4e 45 4e 44 49 53  03
ce90 : 4b 20 46 4f 52 4d 41 54  a4
ce98 : 49 45 52 45 4e 44 49 53  94
cea0 : 4b 20 56 41 4c 49 44 49  6c
cea8 : 45 52 45 4e 20 48 41 55  25
ceb0 : 50 54 4d 45 4e 55 45 20  0b
ceb8 : 20 20 20 20 20 20 a9 00  9e
cec0 : 85 c1 a9 a0 85 c2 a9 c1  3d
cec8 : 85 ae a9 a3 85 af 78 a5  86
ced0 : 01 29 fe 85 01 20 ed f5  8b
ced8 : a5 01 09 01 85 01 58 60  e3
```

Listing 2. Programm »d2« (Schluß)

```
100 REM ***************************             <238>
110 REM *        DRUCKERANPASSUNG        *      <162>
120 REM ***************************             <002>
130 :                                           <106>
140 PRINT"{CLR,4DOWN,12SPACE}DRUCKERANPASS
    UNG"                                        <155>
150 PRINT"{12SPACE}TTTTTTTTTTTTTTTT"            <039>
160 PRINT:PRINT:PRINT                           <029>
170 SYS (57812)"D2",8:POKE 780,0:POKE 781,
    0:POKE 782,192:SYS 65493                    <113>
180 INPUT"GERAETEADRESSE :";GA                  <128>
190 INPUT"{DOWN}SEKUNDAERADRESSE :";SA          <202>
200 INPUT"{DOWN}STEUERCODE FUER BREITSCHRI
    FT :";BR                                    <035>
210 POKE 50855,GA:POKE 49379,GA                 <231>
220 POKE 50857,SA:POKE 49381,SA                 <091>
230 POKE 49426,BR                               <158>
240 OPEN 1,8,15,"S:D2":CLOSE 1                  <025>
250 SYS (57812)"D2",8:POKE 193,0:POKE 194,
    192:POKE 174,224:POKE 175,206               <175>
260 SYS 62957                                   <099>
```

Listing 3. Zur einfachen Anpassung von Polydat an Ihren Drucker können Sie dieses Programm verwenden

PEEKs und POKEs alphabetisch

Die alphabetische Liste der PEEKs und POKEs soll zum schnellen Nachschlagen dienen, wenn Sie eine bestimmte Funktion suchen, die in Basic sonst nicht zu verwirklichen ist.

Die Liste ist in fünf große Gruppen unterteilt, um die speziellen Funktionen schneller finden zu können. Der erklärende Text mit ausführlicher Beschreibung soll Ihnen weiterhin helfen, die aufgeführten PEEKs und POKEs besser nutzen zu können.

Allerdings geht Probieren über Studieren und Sie sollten alles einmal selbst ausprobieren.

Ein-/Ausgabesteuerung allgemein

ASCII-Code der letzten gedrückten Taste (CHR$(0)=keine Taste gedrückt)
PEEK(197)

Bildschirmcode des Zeichens unter dem Cursor:
PEEK(206)

Cursor blinkt schneller
POKE 788,62

Cursorblinken während Programmablauf einschalten
POKE 204,1

Cursorblinken ausschalten
POKE 204,0

Cursor auf angegebene Position setzen (X=Zeile (0 bis 24); Y=Spalte (0 bis 39)).
POKE 211,X:POKE 214,Y

Fehlermeldungsausgabe sperren
POKE 768,61

Fehlermeldungsausgabe wieder zulassen mit
POKE 768,139

Hintergrundfarbe bestimmen (Farbcode: 0 bis 15)
POKE 53281,Farbcode

Rahmenfarbe festlegen
POKE 53280,Farbcode

INPUT-Befehl: Ausgabe des Fragezeichens unterdrücken (Achtung: Cursor springt nach <RETURN> nicht mehr in die nächste Zeile).
POKE 19,1

Normales Verhalten bei INPUT wiederherstellen
POKE 19,0

Invers-Darstellung einschalten
POKE 199,1

Insert-Modus ausschalten (vor jeder GET-Anweisung POKE 216,0 verhindert in Eingaberoutinen etc., daß die Betätigung einer Cursortaste nach <SHIFT INS/DEL> Grafikzeichen erzeugt).
POKE 216,0

Joystickabfrage Port 1 (1 oben; 2 unten; 4 links; 8 rechts; 16 Feuer).
PEEK(56320)

Joystickabfrage Port 2 (1 oben; 2 unten; 4 links; 8 rechts; 16 Feuer).
PEEK(56321)

Länge des Tastaturpuffers ändern
(X=Länge des Puffers=Maximalanzahl der bei zu schnellen Eingaben zur Weiterverarbeitung gespeicherten Tasten)
POKE 649,X

Listing ohne Zeilennummern
POKE 22,35

Repeatfunktion für Tasten (X=0 Normalzustand; X=64 Repeatfunktion für alle Tasten ausschalten; X=128 Repeatfunktion für alle Tasten einschalten).
POKE 650,X

Abfrage der Sondertasten
PEEK(653)
(1 = SHIFT-Taste gedrückt; 2 = CBM-Taste; 4 = CTRL-Taste).

Tastaturpuffer löschen nach jeder GET-Anweisung sorgt dafür, daß im Tastaturpuffer gespeicherte und noch nicht verarbeitete Zeichen gelöscht werden – das »Nachlaufen« von Basic-Programmen bei zu schneller Eingabe wird verhindert.
POKE 198,0

Zeichen in Tastaturpuffer schreiben
POKE 631,ASC("A"):POKE 632,ASC("B")...
ASCII-Codes der Tasten <A>, <B>,... in den Tastaturpuffer schreiben, um entsprechende Tastenbetätigung zu simulieren.

Zeichenfarbe bestimmen (X=Farbcode: 0 bis 15)
POKE 646,X

Zeichensatz-Umschaltung sperren
POKE 657,128

Umschaltung wieder zulassen
POKE 657,0

1. Die Arbeitsweise des Tastaturpuffers zeigt am besten ein kleines Beispiel, wie die folgende »Mini-Textverarbeitung«:

```
100 GET A$
110 FOR I=1 TO 100:NEXT I
120 PRINT A$;
130 GOTO 100
```

Das Demoprogramm wartet auf eine Taste, gibt das zugehörige Zeichen aus und wartet anschließend erneut auf einen Tastendruck. Wegen der Warteschleife in Zeile 110 ist das Programm zu langsam, um Ihren Eingaben zu folgen, wenn Sie sich bemühen, möglichst schnell zu tippen.

Die vom Programm noch nicht verarbeiteten Zeichen speichert der Computer im Tastaturpuffer ab Adresse 631 für die spätere Verarbeitung. In der Speicherzelle 198 wird die Anzahl der noch zu verarbeitenden Zeichen abgelegt, die sich im Tastaturpuffer befinden. Es gehen also keine Zeichen verloren. Wenn der Benutzer schon längst keine Taste mehr betätigt, erscheinen die Zeichen allmählich nach und nach auf dem Bildschirm – das Programm »hinkt hinterher«.

Dieses »Nachlaufen« ist bei Basic-Programmen, die mit GET arbeiten (Textverarbeitung, Eingaberoutine), problemlos zu vermeiden, indem vor jeder GET-Anweisung in Speicherzelle 198 (Anzahl der Zeichen im Tastaturpuffer) der Wert 0 gePOKEt wird. Dem C64 wird damit simuliert, daß keine noch zu verarbeitenden Zeichen im Puffer vorhanden sind, und er wartet auf die nächste Tastenbetätigung. Nachteil: Bei zu schnellen Eingaben gehen die nicht

mehr sofort zu verarbeitenden Zeichen verloren, da sie nicht mehr im Puffer »gerettet« werden.

2. »Simulierter Direktmodus«. Verschiedene Anweisungen sind nur im Direkt-, nicht im Programmodus möglich. Zum Beispiel können Sie nur im Direktmodus Programmzeilen löschen oder ändern.

Der Direktmodus kann jedoch im Programm simuliert werden. Die gewünschte Anweisung wird Zeichen für Zeichen - die ASCII-Codes der Zeichen! - in den Tastaturpuffer ab 631 gePOKEt. In 198 wird die Zeichenanzahl gePOKEt und das Programm mit END beendet. Nach dem Beenden eines Programms arbeitet der C64 automatisch die Zeichen im Tastaturpuffer ab, also die von Ihnen dort abgelegten Tasten.

```
100 INPUT"FUNKTION (BSP. Y=2*COS(X))";A$
110 PRINT CHR$(147);:REM BILDSCHIRM LOESCHEN
120 PRINT "210 ";A$:REM FUNKTION IN OBERSTE ZEILE
130 PRINT "RUN 200":REM PROGRAMM AB ZEILE 200
    STARTEN
140 POKE 631,19:REM CODE VON 'CURSOR HOME'
150 POKE 632,13:REM RETURN-CODE
160 POKE 633,13:REM RETURN-CODE
170 POKE 198,3:REM SIMULATION VON 3
    TASTENBETAETIGUNGEN
180 END
190
200 INPUT "X-WERT";X
210 Y=SIN(X):REM DIESE ZEILE WIRD GEAENDERT !!!
220 PRINT Y
230 GET A$:IF A$="" THEN 230:REM AUF TASTE WARTEN
240 GOTO 100:REM NEUE FUNKTION
```

Wenn Sie dieses Demoprogramm eingeben und starten, werden Sie nach einer Funktion gefragt. Geben Sie zum Beispiel ein Y=2*X, und drücken Sie <RETURN>.

Der Bildschirm wird gelöscht und in der obersten Zeile die Zeilennummer 210 und dahinter Ihre Funktionsvorschrift ausgegeben. In der folgenden Zeile wird die Anweisung RUN 200 ausgegeben

200 Y=SIN(X)
RUN 200

In den Tastaturpuffer werden nacheinander die Codes der Tasten <CURSOR HOME>, <RETURN> und nochmal <RETURN> gePOKEt, in den »Zeichenzähler« 198 entsprechend die Zahl 3.

Nach der END-Anweisung bearbeitet der Computer die im Tastaturpuffer abgelegten Zeichen. Das erste Zeichen - CURSOR HOME - setzt den Cursor auf die oberste Bildschirmzeile. Das zweite Zeichen - RETURN - sorgt für die Ausführung der Anweisung, also die Übernahme der neuen Funktion als Zeile 210.

Der zweite RETURN-Code bewirkt die Ausführung der nächsten Anweisung: RUN 200. Das Programm wird ab Zeile 200 neu gestartet. Es fragt Sie nach einem X-Wert, berechnet mit der geänderten Funktionsvorschrift in Zeile 210 den zugehörigen Y-Wert und gibt ihn aus. Wenn Sie anschließend eine beliebige Taste drücken, geht dieses Spiel von vorne los.

Grafik

(VIC=Speicherzelle 53248)
Grafikmodus
Einschalten: POKE VIC+17,PEEK(VIC+17) OR 32
Ausschalten: POKE VIC+17,PEEK(VIC+17) AND 223

Grafikspeicher
Grafikspeicher-Startadresse ab Adresse 8192: POKE VIC+24,PEEK(VIC+24) OR 8

Grafik löschen
Grafikspeicher löschen (Voraussetzung: Beginn ab 8192)
FOR I=8192 TO 8192+8000:POKE I,0:NEXT I

Multicolor-Modus
Einschalten (Voraussetzung: Grafikmodus wurde zuvor eingeschaltet) POKE VIC+22,PEEK(VIC+22) OR 16

1. Grafikspeicher schützen. Wie Sie die Hires-Grafik ein-/ausschalten, den Beginn des Grafikspeichers auf Adresse 8192 legen und den Grafikspeicher löschen, finden Sie in der zugehörigen Abteilung unserer Liste.

Anschließend liegt der Grafikspeicher jedoch mitten im eigentlich für das Basic-Programm und die Variablen verwendeten Speicherbereich. Eine Möglichkeit, dieses Problem zu lösen, besteht darin, den für Basicprogramm verfügbaren Speicherbereich zu begrenzen und dem C64 mitzuteilen, daß er bei Adresse 8191 endet:

POKE 5,255:POKE 56,31

Diese beiden Befehle sollten sich in der ersten Zeile Ihres Grafikprogramms befinden! Ihnen stehen nun zwar nur noch 7 KByte für Ihr Programm zur Verfügung, zum Experimentieren mit der Grafik reicht das jedoch allemal.

2. Punkte setzen/löschen: Die »Grafik-POKEs« übernehmen zwar alle notwendigen Vorbereitungen, der eigentliche Sinn von Grafikprogrammen, das Setzen oder Löschen von Punkten, fehlt jedoch in der Liste. Die folgenden beiden Formeln setzen voraus, daß den Variablen X und Y die Punktkoordinaten zugewiesen wurden.

AD = 320 * INT(Y/8) + (Y AND 7) + 8*INT(X/8)
BN = 7 - (X AND 7)

Nach diesen Berechnungen kann der betreffende Punkt gesetzt oder gelöscht werden.

Setzen: POKE 8192+AD,PEEK(8192+AD) OR 2↑BN
Löschen: POKE 8192+AD,PEEK(8192+AD) AND (255-2↑BN)

Die obigen Berechnungen funktionieren jedoch nur, wenn die Hires-Grafik mit 320 x 200 Punkten eingeschaltet ist. AD ist dabei die Nummer des Bytes ab der Anfangsadresse des Grafikbildschirms und BN ist die Nummer des Bits in dem Byte Anfangsadresse + AD.

Sprites

(VIC=53248, NR=Spritenummer (0 bis 7))
Breite verdoppeln
POKE VIC+29,PEEK(VIC+29) OR 2↑NR

Höhe verdoppeln
POKE VIC+23,PEEK(VIC+23) OR 2↑NR

Kollision feststellen
PEEK(VIC+30):POKE VIC+30,0
(die Bits der beiden kollidierten Sprites sind gesetzt; in VIC+30 muß (!) anschließend der Wert 0 gePOKEd werden, da diese »Kollisionsspeicherzelle« nicht automatisch gelöscht wird!).

Multicolor-Sprite
Einschalten: POKE VIC+28,PEEK(VIC+28) OR 2↑NR

Spritefarben
Definieren: POKE VIC+39+NR,Farbcode

Sprites einschalten
POKE VIC+21,PEEK(VIC+21) OR 2↑NR

Sprites ausschalten
POKE VIC+21,PEEK(VIC+21) AND 255-2↑NR.

X-/Y-Koordinaten
Festlegen durch POKE VIC+2*NR,X:POKE VIC+2*NR+1,Y

Die »Sprite-POKEs« werden verständlicher, wenn Sie ein wenig über den Umgang des C64 mit Sprites erfahren. Bis zu acht Sprites können gleichzeitig verwaltet werden. Jedes Sprite bekommt eine Nummer zwischen 0 und 7. Für eine Sprite-Funktion ist meist ein Register des VIC-Chips zuständig, zum Beispiel Register 29 für die Verdoppelung der Sprite-Höhe. Um ein Register anzusprechen, müssen Sie die Registernummer zur Basisadresse des VIC-Chips addieren:
53248 (Basisadresse) + 29 (Höhen-Register) = 53277

Die Speicherzelle 53277 beeinflußt also die Sprite-Höhe. Um gezielt ein bestimmtes Sprite ansprechen zu können, ist bei fast allen für die Sprites zuständigen Registern je ein Bit einem Sprite zugeordnet.
Bit 0 Sprite Nummer 0
Bit 1 Sprite Nummer 1
...
...
Bit 7 Sprite Nummer 7

Um nun gezielt die Höhe von Sprite Nummer 5 zu verdoppeln, ohne auch alle anderen Sprites zu beeinflussen, muß Bit 5 von Register 53248 gesetzt werden. Gesetzt beziehungsweise gelöscht werden Bits mit Hilfe der logischen Operatoren OR (Setzen) und AND (Löschen). Das folgende Schema zeigt, wie einzelne Bits gezielt manipuliert werden
Bit NR setzen POKE X,PEEK(X) OR 2↑NR
Bit NR löschen POKE X,PEEK(X) AND 255-(2↑NR)

Zur Verdoppelung der Höhe eines Sprites finden Sie in der Tabelle den POKE:
POKE VIC+29,PEEK(VIC+29) OR 2↑NR

NR ist hierbei die Spritenummer (0 bis 7), VIC die Basisadresse des VIC-Chips (53248). Um die Höhe von Sprite 3 zu verdoppeln, setzen Sie einfach das entsprechende Bit Nummer 3 mit der Anweisung
POKE 53248+29,PEEK(53248+29) OR 2↑3

Die Verdoppelung können Sie jederzeit rückgängig machen, indem Sie dieses Bit wieder löschen
POKE 53248+29,PEEK(53248+29) AND 255-2↑3

Dieser kleine »Ausflug« sollte verdeutlichen, wie die Inhalte der Sprite-Register vom C64 interpretiert werden. Zum Umgang mit Sprites halten Sie sich bitte einfach an die POKE-Liste. Die Anwendung dürfte problemlos sein, nachdem Ihnen nun die Bedeutung der Variablen VIC und NR bekannt ist.

Sound

(SID=54272; alle Angaben beziehen sich auf Stimme 1; für Stimme 2 zur angegebenen Adresse den Wert 7 addieren, für Stimme 3 den Wert 14).

Gesamtlautstärke:
Für alle Stimmen POKE SID+24,X (X zwischen 0 und 15).

Hüllkurve:
DECAY- und ATTACK-Wert festlegen POKE SID+5,X
SUSTAIN- und RELEASE-Wert festlegen POKE SID+6,X

Puls-Pause-Verhältnis
POKE SID+2,LB:POKE SID+3,HB (LB/HB Low- und High-Byte der Pulsbreite).

Schwingungsform
POKE SID+4,PEEK(SID+4) OR 2↑I
(I=4→Dreieck; I=5→Sägezahn; I=6→Rechteck; I=7→Rauschen).

Tonhöhe:
POKE SID,HB:POKE SID+1,LB
HB/LB High- und Low-Byte der gewünschten Tonhöhe (siehe Tabelle im Handbuch).

Programmschutz

Ändern von Programmzeilen verhindern
POKE 813,2

Basic-Programm zerstören
POKE 776,1

LIST sperren
POKE 775,200

LIST wieder zulassen
POKE 775,167

Rechner nimmt keine Befehle mehr an
POKE 120,2

RESET nach Programmende
POKE 768,143

RESET bei LIST-Anweisung
POKE 774,226:POKE 775,252

RESET bei Drücken der RESTORE-Taste
POKE 792,226:POKE 793,252

RESET bei SAVE-Anweisung
POKE 818,226:POKE 818,165

Speichern verhindern
POKE 801,0:POKE 802,0:POKE 818,165

Sperren von Tastatureingaben
POKE 649,0

Wiederzulassen von Tastatureingaben
POKE 649,10

STOP-Taste ausschalten
POKE 788,52

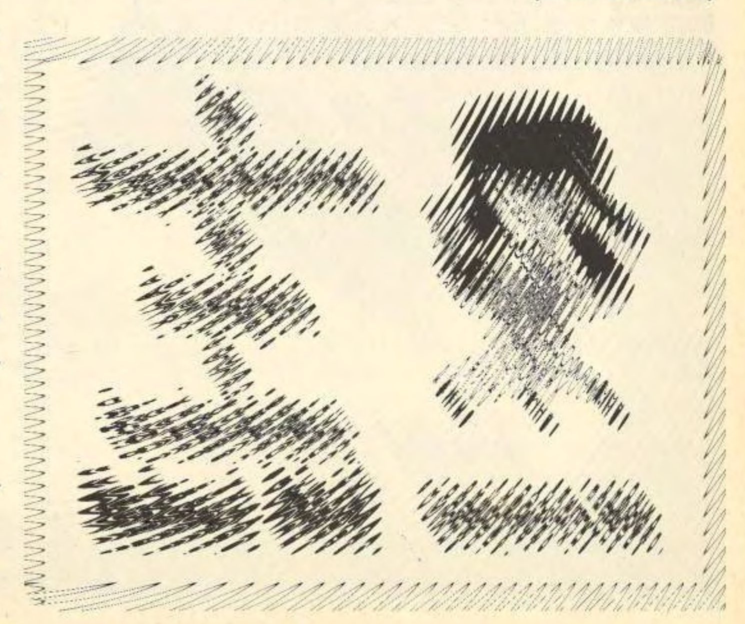
STOP-Taste wieder einschalten
POKE 788,49

STOP- und RESTORE-Taste ausschalten
POKE 792,193

STOP- und RESTORE-Taste wieder einschalten
POKE 792,71

Die angegebenen POKEs sind Ihnen sicher immer wieder von Nutzen bei der Programmierung, wobei es egal ist, ob Sie Basic oder Assembler verwenden. Bei manchen POKEs ist jedoch Vorsicht geboten, da sie das vorhandene Programm zerstören können. Am besten, Sie testen die verschiedenen POKEs vor dem Einsatz im Programm, um böse Überraschungen zu vermeiden.

(Said Baloui/rb)



Wie von Geisterhand...

Beim C64 gibt es die Möglichkeit, Programme nach dem Laden automatisch starten zu lassen. Sie ersparen sich so die Eingabe von RUN.

Wenn ein Programm mit einem Autostart versehen wurde, genügt die Eingabe von LOAD "Name",8,1, um das Programm automatisch laden und starten zu lassen. Dazu wird auf der Diskette ein kleines Programm erzeugt, das automatisch startet, dann Ihr eigentliches Programm nachlädt und wiederum startet. Als erstes müssen Sie aber das Listing 1 mit Hilfe des MSE (siehe Seite 159) abtippen und auf einer Diskette speichern. Zur Anwendung gehen Sie so vor:

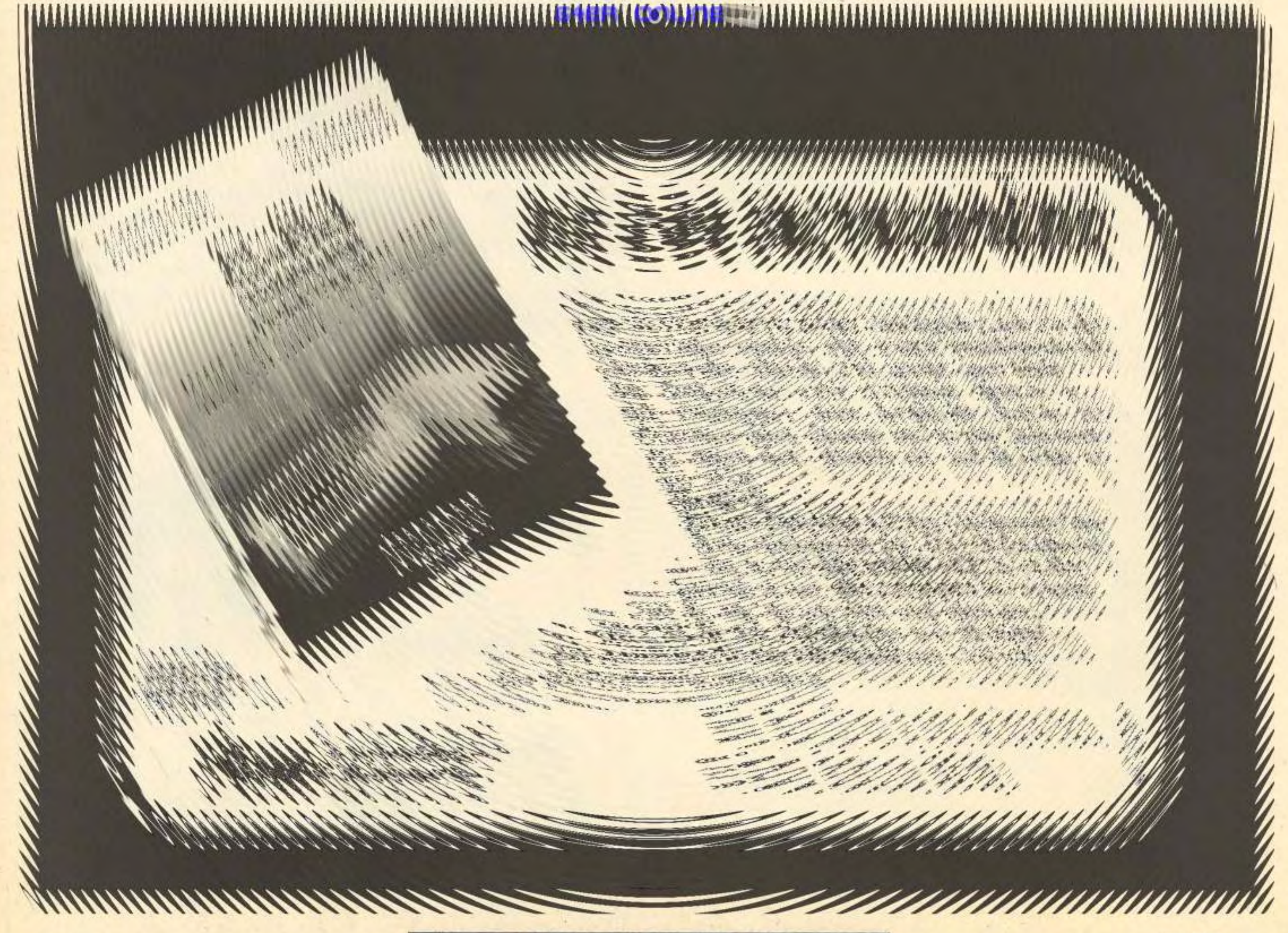
1. LOAD "AUTOSTART",8,1 : NEW
2. Laden Sie jetzt das Programm, das Sie mit einem Autostart versehen wollen. Einzige Bedingung: Es muß sich mit RUN starten lassen.
3. Überlegen Sie sich jetzt zwei Programmnamen: einen für das Ladeprogramm und einen für das neue Hauptprogramm (zum Beispiel PACMAN und PAC.MAIN).
4. Bei diesem Autostart haben Sie zusätzlich die Möglichkeit, Ihr Programm auf Diskette verschlüsseln zu lassen. Suchen Sie sich daher noch einen beliebigen Code zwischen 1 und 255 aus. Wenn Sie Ihr Programm nicht verschlüsseln wollen, wählen Sie Code 0.
5. SYS 49152,Code, "Hauptprogrammname", "Ladername" (Beispiel: SYS 49152,200, "PAC.MAIN", "PACMAN")
6. Ihr Programm wird nun neu auf Diskette gespeichert.

Fertig! Wenn Sie nun LOAD "Ladername",8,1 (also zum Beispiel LOAD "PACMAN",8,1) eingeben, wird PACMAN automatisch geladen und gestartet. Zusätzlich wird Ihr Programm noch mit einem Schutz gegen Drücken von <RUN/STOP RESTORE> versehen.

(Christoph Dautzenberg/tr)

```
Name : autostart              c000 c131
----------------------------------------
c000 : 20 fd ae 20 9e b7 8e ac  0a
c008 : c0 20 fd ae 20 9e ad 20  1b
c010 : a3 b6 8d 6f c0 c9 0d 90  0f
c018 : 03 4c 71 a5 20 bd ff 20  83
c020 : 11 c1 a2 08 86 ba 20 35  e5
c028 : c0 a9 2b a6 2d a4 2e 20  4d
c030 : d8 ff 4c bf c0 a5 2b 85  04
c038 : fb a5 2c 85 fc a0 00 b1  fa
c040 : fb 4d ac c0 91 fb c8 d0  e3
c048 : f6 a5 fc e6 fc c5 2e d0  85
c050 : ee 60 a2 ea 8e 28 03 bd  26
c058 : 77 02 4d 00 03 9d 80 7f  42
c060 : ca 30 f4 a2 04 bd 10 fd  3e
c068 : 9d 04 80 ca 10 f7 a9 0c  00
c070 : a2 0d a0 80 20 bd ff a9  14
c078 : 00 85 9d 20 d5 ff 86 2d  78
c080 : 98 a6 2b 86 fb a4 2c 84  a6
c088 : fc 20 57 a6 a8 b1 fb 4d  e2
c090 : 00 03 91 fb c8 d0 f6 a5  30
c098 : 2e e7 fc 10 f0 20 53 e4  22
c0a0 : 4c ae a7 e2 fc 5e fe 43  cf
c0a8 : 48 44 38 36 00 45 a6 02  b0
c0b0 : 28 43 29 38 36 20 42 59  eb
c0b8 : 20 43 48 44 4c 79 00 20  e5
c0c0 : 35 c0 20 79 00 c9 2c d0  2d
c0c8 : f3 20 73 00 20 9e ad 20  96
c0d0 : a3 b6 c9 00 d0 05 a2 08  11
c0d8 : 4c 37 a4 20 bd ff a2 52  f8
c0e0 : a0 c0 86 ac 84 ad a2 bc  d1
c0e8 : a0 c0 86 ae 84 af a9 61  8f
c0f0 : 85 b9 20 d5 f3 20 8f f6  81
c0f8 : a9 08 20 b1 ff a9 61 20  f7
c100 : 93 ff a9 a6 20 dd ed a9  ce
c108 : 02 20 dd ed a0 00 4c 24  d3
c110 : f6 86 fb 84 fc a8 88 b1  73
c118 : fb 4d ac c0 99 b0 c0 88  30
c120 : 10 f5 a0 03 b9 a3 c0 4d  0a
c128 : ac c0 99 a3 c0 88 10 f4  8a
c130 : 60 00 00 00 00 00 00 00  91
```

Listing 1. »Autostart« bitte mit dem MSE eingeben



Wie gebe ich Programme ein?

Es ist zweifellos eine faszinierende Sache: Ein Programm abzutippen und dann das Ergebnis zu bewundern. Hier erfahren Sie, was Sie beim Eintippen von Listings beachten müssen und mit welchen Tricks Sie zum »Tipp-Profi« werden.

Jeder frischgebackene Computer-Besitzer kommt einmal an den Punkt, an dem er erkennt, daß das »Elektronik-Wunder« C 64 ohne entsprechende Fütterung (sprich Programme) so dumm wie ein kleiner Taschenrechner ist. Dann gibt es zwei Möglichkeiten: Entweder man kauft sich fertige Programme für teures Geld, oder man geht zum nächsten Zeitschriftenhändler und besorgt sich das 64'er-Magazin. Da Sie den zweiten Weg gewählt haben (sonst würden Sie diesen Artikel nicht lesen) möchten wir Ihnen gratulieren. Zweifellos verspüren Sie beim Lesen der Programmbeschreibungen sofort das Bedürfnis, diese herrlichen Programme abzutippen.

Daraufhin folgt die Ernüchterung: »...verwenden Sie zur Eingabe bitte den MSE...« oder »...bitte mit dem Checksummer eingeben...«. Was sind denn das für Wunderprogramme? Kann man die irgendwo kaufen? Nein, der Checksummer und der MSE sind von uns entwickelte Programme, die Ihnen helfen sollen, unsere Listings fehlerfrei abzutippen (als »Listing« wird im folgenden ein abgedrucktes Programm bezeichnet). Denn ein fehlerfreies Abtippen ist die Voraussetzung für ein korrektes Funktionieren des Programms.

Also: Ihr erster Schritt soll sein, den Checksummer und den MSE auf Seite 159 abzutippen. Doch wie macht man das eigentlich?

Das Allerwichtigste, was Sie sich einprägen müssen, ist die Bedeutung der RETURN-Taste (ganz rechts auf der Tastatur des C 64): Wenn Sie auf der Tastatur etwas eingeben, so ist das in etwa so, als ob Sie dem Computer einen Brief schreiben. Die Bildschirmzeile(n), auf der Sie gerade Buchstaben und Zeichen eintippen, ist das Briefpapier. Der Computer kümmert sich erst dann um das, was Sie schreiben, wenn Sie die RETURN-Taste drücken. Sie können das etwa damit vergleichen, daß Sie den eben geschriebenen »Brief« durch die RETURN-Taste an den Computer »abschicken«. Erst dann versucht der C 64 das, was Sie ihm mitteilen wollen, zu verstehen, zu »interpretieren«. Also: Nach einer Eingabe immer <RETURN> drücken.

Das Zweitwichtigste, was Sie sich merken müssen, ist, daß der C 64 keine »Briefe« annimmt, die länger als zwei Bildschirmzeilen sind! Das bedeutet: Wenn Sie etwas eintippen, und Sie überschreiten mit dem »Cursor« (der blinkenden Schreibmarkierung) das Ende der zweiten Zeile, dann »vergißt« der C 64 alles, was Sie in den beiden vorhergehenden Zeilen geschrieben haben. Er betrachtet dann nach dem Drücken der RETURN-Taste die dritte Zeile als das aktuelle »Briefpapier«. Das ist ganz besonders wichtig, wenn Sie ein Basic-Programm (mit den Zahlen (= »Zeilennummern«) am Anfang jeder Zeile) eintippen. Sobald der Cursor in der dritten Schreibzeile steht, wird es Probleme geben. Also: Beim Eintippen von Listings darauf achten, daß der gerade eingegebene Text nicht länger als zwei Bildschirmzeilen ist. Falls es nicht anders geht, können Sie die Freiräume zwischen den einzelnen Basic-Befehlen ersatzlos weglassen (diese Freiräume werden übrigens auch »Spaces« genannt). Unsere Listings enthalten diese Spaces nur, weil das Programm dadurch wesentlich übersichtlicher wird.

Basic-Programme

Nun wird es interessant: Sie beginnen, ein Basic-Listing abzutippen. Wichtig ist jetzt vor allem, daß Sie die Zeilennummern am Anfang jeder Programmzeile nicht vergessen. Wenn Sie beim Eintippen auf geschweifte Klammern stoßen ({CLR}) dürfen Sie das, was innerhalb der Klammern steht, nicht mit eintippen! Statt dessen drücken Sie die dort angegebenen Tasten. Also: In einem unserer Listings steht zum Beispiel:

```
10 PRINT "{CLR,3DOWN}HALLO!"
```

Beim Eintippen müssen Sie folglich nach dem Anführungszeichen zuerst die Taste <CLR> drücken. Wenn Sie diese Taste betrachten, wird Ihnen auffallen, daß die CLR-Funktion oben auf der Taste steht. Das bedeutet, daß Sie diese Taste zusammen mit der SHIFT-Taste drücken müssen, um die CLR-Funktion zu erreichen. Auf dem Bildschirm muß jetzt nach dem Anführungszeichen ein dunkles Herz auf hellem Grund erscheinen. Man nennt diesen Darstellungsmodus auch »revers« oder »invers«. Als nächstes müssen Sie dreimal die »Down«-Taste drücken. Das ist die »Cursor-nach-unten«-Taste. Neben dem inversen Herzchen müssen jetzt drei inverse Qs erscheinen.

```
<249>        CHECKSUMMER 64 V3

          EINEN MOMENT, BITTE ...


          CHECKSUMMER AKTIVIERT.

AUSSCHALTEN : POKE1,55
ANSCHALTEN  : POKE1,53
READY.
10 PRINT "HALLO"
```

»Checksummer«, die Basic-Eingabehilfe. Links oben sehen Sie die Prüfsumme für die Programmzeile unten.

```
 programm : mse                0801 0eaf

   startadresse : $0801
0801 : 0b 08 0a 00 9e 32 30 36 3c
0809 : 31 00 00 00 a2 08 a9 36 b8
0811 : 85 a4 a9 08 85 a5 a9 00 80
0819 : 85 a6 a9 b0 85 a7 a0 00 8a
0821 : b1 a4 91 a6 c8 d0 f9 e6 26
0829 : a5 e6 a7 ca d0 f2 a9 36 3c
0831 : 85 01 4c 00 b0 20 d1 b1 01
0839 : a9 06 8d 21 d0 a9 03 8d ee
0841 : 20 d0 8d 86 02 a0 b3 a9 45
0849 : 74 20 ff b1 a0 b3 a9 b9 c5
0851 : 20 ff b1 a0 00 20 cf ff 31
0859 : 99 01 02 c8 c9 0d d0 f5 41
0861 : 88 f0 d2 c0 0f 90 02 a0 ed
0869 : 0e 8c 00 02 20 ea b1 a0 5f
0871 : b3 a9 cf 20 ff b1 20 8e 1c
```

Das MSE-Programm dient der leichten und fehlerfreien Eingabe von Maschinensprache-Listings.

Wir wollen nun untersuchen, was passiert, wenn Sie dieses Mini-Programm mit »RUN« starten: Der C 64 »liest« den PRINT-Befehl und erkennt, daß er den folgenden Text auf dem Bildschirm ausgeben soll. Als erstes soll er den Bildschirminhalt löschen (CLR) und dann drei Zeilen weiter unten (3DOWN) den Text »HALLO« ausdrucken. Wir haben nun schon einen wichtigen Punkt gelernt: Sobald wir ein Anführungszeichen eingeben, werden alle »Steuerfunktionen« (Cursor-Bewegungen, Bildschirm löschen, die Farbwahl-Tasten etc.) nicht mehr direkt ausgeführt, sondern als »Steuerzeichen« in den Programmtext übernommen. Dieser besondere Modus wird »Anführungszeichen-Modus« genannt.

Was aber tun, wenn wir uns innerhalb von Anführungszeichen vertippt haben und dringend die »Cursor-nach-links«-Taste zum Korrigieren benötigen? Der C 64 würde diese Tasten-Funktion ja nicht ausführen, sondern das entsprechende Steuerzeichen ausgeben. Sie können das leicht ausprobieren, wenn Sie einfach mal ein Anführungszeichen eingeben, und dann versuchen, mit dem Cursor ein Zeichen nach links zu fahren. Aus diesem Anführungszeichen-Modus führen zwei Wege nach draußen:

1. Wir geben noch ein Anführungszeichen ein und schalten dadurch den Modus aus. Das eventuell störende zweite Anführungszeichen können wir ja wieder löschen. Hauptsache, der Cursor macht wieder das, was wir wollen.
2. Wir drücken <SHIFT> und <RETURN>. Dadurch wird der Cursor an den Anfang der nächsten Zeile gesetzt, ohne daß das gerade Eingegebene vom C 64 beachtet wird. Als Nebeneffekt des Ganzen wird der Anführungszeichenmodus wieder gelöscht und der Cursor darf sich wieder frei bewegen.

Steuerzeichen

Als kleine Übung können Sie ja einmal das Listing 1 abtippen. Es zeigt, wie man nur durch die Cursor-Steuerzeichen und einigen Strichen eine einfache, bewegte Grafik auf den Bildschirm zaubern kann.

Doch nun zu ein paar allgemeinen Hinweisen:

1. Wie löscht man eine Programmzeile? Ganz einfach: Wir geben nur die Nummer der fehlerhaften Zeile ein und drücken die RETURN-Taste.

2. Wenn Sie ein Programm fertig eingetippt haben, ist das Allerwichtigste, daß Sie Ihre Arbeit unbedingt zuerst auf Diskette oder Kassette speichern! Denn wenn Sie noch irgendwo einen gravierenden Tippfehler gemacht haben, kann dies zum rettungslosen Absturz des Computers führen. Und dann hilft nur noch das für Ihr Programm tödliche Aus- und wieder Einschalten des Computers. Also: Programme vor dem ersten Testlauf immer speichern!

3. Was ist beim Speichern auf Diskette oder Kassette zu beachten? Bei Diskette ist es wichtig, daß Sie nach dem SAVE-Befehl einen Programmnamen (maximal 16 Zeichen) angeben. Ansonsten meldet der Computer einen »missing filename error«. Bei Datasette können Sie den Programmnamen weglassen. Im Fachjargon wird übrigens ein Programm als »File« bezeichnet. Deswegen werden Sie häufig auch »Filename« statt »Programmname« lesen.

4. Wenn in einem Basic-Programm Zeilen vorkommen, die vom Inhalt her identisch oder zumindest stark ähnlich sind, gibt es einen kleinen Trick: Um sich Tipparbeit zu sparen, genügt es, wenn wir die Zeile einmal auf den Bildschirm schreiben, die RETURN-Taste drücken und dann nur noch die Zeilennummer durch die neue überschreiben, um die

```
10 PRINT"{CLR}":A$="{HOME,11DOWN,18RIGHT}" <049>
20 PRINT A$"{DOWN,SPACE,UP,LEFT}M{DOWN,RVS
   ON,SPACE,RVOFF}"                        <143>
30 GOSUB 200                               <238>
40 PRINT A$" ="                            <166>
50 GOSUB 200                               <002>
60 PRINT A$"{RIGHT,SPACE}N"                <002>
70 GOSUB 200                               <022>
80 PRINT A$"{2RIGHT,SPACE,DOWN,LEFT}*"     <123>
90 GOSUB 200                               <042>
100 PRINT A$"{2RIGHT,DOWN,SPACE,DOWN,LEFT}
    M"                                     <104>
110 GOSUB 200                              <062>
120 PRINT A$"{2RIGHT,2DOWN,SPACE,2LEFT}="  <249>
130 GOSUB 200                              <082>
140 PRINT A$"{2DOWN,RIGHT,SPACE,2LEFT}N"   <180>
150 GOSUB 200                              <102>
160 PRINT A$"{2DOWN,SPACE,UP,LEFT}*"       <066>
170 GOSUB 200                              <122>
180 GOTO 20                                <110>
200 FOR I=1 TO 100:NEXT:RETURN             <042>
```

© 64'er

Listing 1. Ein Beispiel für Steuerzeichen in Basic-Listings. Bitte beachten Sie die Hinweise im Text.

anderen Zeilen zu erhalten. Ein Beispiel: Betrachten wir uns in Listing 1 die Zeilen 30, 50, 70 und so weiter. Alle haben den gleichen Inhalt. Es genügt, wenn wir die Zeile einmal auf den Bildschirm schreiben, die RETURN-Taste drücken und dann einfach die Zeilennummer 30 durch eine 50, 70, 90 etc. überschreiben (<RETURN> nicht vergessen). Wenn wir jetzt »LIST« eingeben, sehen wir, daß der C 64 alle Zeilen angenommen hat. Dies ist eine äußerst effektive Methode, die man sich unbedingt merken sollte.

So, Sie haben jetzt (fast) alles gelernt, was zum fehlerfreien Abtippen eines Programms wichtig ist. Man könnte nun noch sämtliche Fehlermeldungen des C 64 aufzählen und zu jeder Meldung mögliche Fehlerquellen, Tips zum Beheben und so weiter geben. Das würde jedoch den Rahmen dieses Artikels bei weitem sprengen!

Das Eintippen von Maschinensprache-Programmen ist dank unseres MSE-Programms auf Seite 159 schon deutlich einfacher. Wenn Sie die Bedienungsanleitung zum MSE auf Seite 159 genau beachten, sind Tippfehler fast ausgeschlossen. Wenn Sie ein solches MSE-Listing vollständig eingetippt und gespeichert haben, sind lediglich einige Regeln zu beachten:

1. Maschinensprache-Programme müssen immer (!) mit dem Zusatz »,1« zum LOAD-Befehl geladen werden. Beispiel:
LOAD"LISTING",8,1 (Diskette) oder LOAD"LISTING",1,1 (Kassette). Man nennt dies »absolut« laden. Es hat etwas mit der Stelle im Speicher zu tun, an die das Programm geladen werden soll.
2. Maschinensprache-Programme werden fast immer durch einen SYS-Befehl und nicht durch »RUN« gestartet. Es gibt eine Ausnahme:
Manche Programme haben eine Basic-Zeile mit dem entsprechenden SYS-Befehl zum Starten. Diese Programme werden mit »RUN« aktiviert. Sie erkennen sie an der Startadresse »0801« im MSE-Listing.
3. Bei Maschinensprache-Programmen ist es ganz besonders wichtig, daß nicht zwei Programme gleichzeitig im selben Speicherbereich liegen dürfen. Der jeweils verwendete Bereich steht in der Kopfzeile des MSE-Programms.
4. Die Startadresse hinter dem SYS-Befehl ist für das Programm »lebenswichtig«. Wenn Sie sich hier vertippen, kann unter Umständen der Computer vollständig abstürzen.

Damit wären wir am Ende dieser kleinen »Einführung in die Kunst des Programme-Abtippens« angelangt. Wir hoffen, daß wir Ihnen in diesem Artikel eine Menge nützlicher Hinweise geben konnten. Und verzagen Sie nicht: Jeder »syntax error« hat einen auffindbaren Grund! (tr)

Checksummer V3 und MSE

Diese beiden Programme sind unentbehrlich beim Abtippen unserer Listings. Sie helfen Tippfehler vor allem bei Maschinenprogrammen zu vermeiden und sparen eine Menge Zeit.

Nobody is perfect. Jeder Computer-Fan, egal ob blutiger Anfänger oder ausgefuchster Profi, macht beim Abtippen von Programmen Tippfehler. Diese Fehler später zu finden, kann ein langwieriges Unterfangen sein.

Deshalb haben wir für Sie die Programme »Checksummer V3« und »MSE« (MaschinenSpracheEditor) entwickelt. Der Checksummer ist für Basic-Programme und der MSE für Maschinensprache-Listings zuständig.

Der Checksummer

Zuerst einmal müssen Sie das Checksummer-Programm (siehe Listing 1) abtippen. Dabei sollten Sie äußerst sorgfältig vorgehen, vor allem bei den Zahlen in den DATA-Zeilen 20 bis 30. Wenn Sie trotzdem noch einen Tippfehler gemacht haben, meldet sich das Programm später mit einem entsprechenden Hinweis. Wenn Sie fertig sind, speichern Sie das Programm auf Diskette oder Kassette.

Jetzt geht es los:

1. Starten Sie den Checksummer durch die Eingabe von »RUN« und das Drücken der RETURN-Taste.
2. Wenn die Meldung »Checksummer aktiviert...« auf dem Bildschirm erscheint, haben Sie keinen Tippfehler gemacht und der Checksummer ist nun eingeschaltet.
3. Zum Löschen des Basic-Programms geben Sie bitte »NEW« ein. Keine Angst, der Checksummer selbst wird dadurch nicht gelöscht.
4. Nun können wir den Checksummer testen. Geben Sie bitte folgende Zeile ein und drücken Sie die RETURN-Taste:

1 REM

In der linken oberen Bildschirmecke sehen Sie nun die Prüfsumme über die eben eingegebene Basic-Zeile. Sie muß <63> lauten. Dem Checksummer ist es übrigens egal, ob Sie »1 REM« oder »1REM« eintippen. Nur innerhalb von Anführungszeichen ist die richtige Anzahl an Leerzeichen wichtig. Diese Prüfsummen erscheinen (sofern Sie den Checksummer eingeschaltet haben) immer dann, wenn Sie eine Basic-Zeile eintippen und dann die RETURN-Taste drücken. In der 64'er finden Sie die Prüfsumme immer am Ende jeder Programmzeile.

```
10 PRINT"CHECKSUMMER FUER C 64"
11 PRINT:PRINT"EINEN MOMENT, BITTE ..."
12 FOR I=828 TO 864:READ A:POKE I,A:PS=PS+
   A:NEXT I
13 IF PS<>5765 THEN PRINT"TIPPFEHLER IN DE
   N ZEILEN 20 BIS 22":END
14 SYS 828:PS=0:FOR I=58464 TO 58583:READ
   A:POKE I,A:PS=PS+A:NEXT I
15 IF PS<>16147 THEN PRINT"TIPPFEHLER IN D
   EN ZEILEN 22 BIS 30":END
16 POKE 1,53:POKE 42289,96:POKE 42290,228
17 PRINT"CHECKSUMMER AKTIVIERT."
18 PRINT:PRINT" AUSSCHALTEN : POKE1,55 ODE
   R"SPC(27)"<RUN/STOP+RESTORE>"
19 PRINT:PRINT" ANSCHALTEN : POKE1,53"
20 DATA 169,0,133,254,162,1,189,93,3,133,2
   55,160,0,177,254
21 DATA 145,254,136,208,249,230,255,165,25
   5,221,95,3,208,238,202
22 DATA 16,230,96,160,224,192,0,160,2,169,
   0,170,133,254,177
23 DATA 95,240,40,201,32,208,3,200,208,245
   ,133,255,138,41,7
24 DATA 170,240,14,72,165,255,24,42,105,0,
   202,208,249,133,255
25 DATA 104,170,232,165,255,24,101,254,133
   ,254,76,111,228,192,4
26 DATA 48,219,198,214,165,214,72,162,3,16
   9,32,157,1,4,189
27 DATA 212,228,32,210,255,208,12,0,92,72,
   32,201,255,170,104
28 DATA 144,1,138,96,202,16,228,166,254,16
   9,0,32,205,189,169
29 DATA 62,32,210,255,104,133,214,32,108,2
   29,169,141,32,210,255
30 DATA 76,128,164,9,60,18,19
© 64'er
```

Listing 1. Der »Checksummer 64 V3« für Basic-Listings

```
5 PRINT CHR$(14)                                  <242>
10 PRINT"{CLR}"                                   <254>
20 PRINT"Q*****************************W"         <130>
30 PRINT"{4DOWN,2SPACE}TEST{SPACE,BLUE,6SP
   ACE}"                                          <022>
40 PRINT"BBBBBBBBBBBBBBBBBBBBBBBBBBBBB"           <108>
© 64'er
```

Bild 1. Die Bedeutung der Steuerzeichen wird im nachfolgenden Text erklärt

In Zeile 10 müssen Sie nach den Anführungsstrichen die Tasten <SHIFT CLR/HOME> drücken und nicht die Klammern mit dem Wort CLR eingeben. In Zeile 20 drücken Sie nach den Anführungsstrichen die CBM-Taste und den Buchstaben <Q>, gefolgt von mehreren SHIFT- und Stern-Tasten und zum Schluß die CBM-Taste und den Buchstaben <W>. In Zeile 30 ist es viermal die CURSOR-abwärts-Taste, gefolgt von zweimaliger Leertaste, dann <SHIFT T> und normal EST, zum Schluß noch einmal die Leertaste, die Farbtaste Blau <CTRL 7> und sechsmal die Leertaste. Zeile 40 besteht lediglich aus mehreren Grafikzeichen, die mit der CBM-Taste und <B> erzeugt werden.

CTRL steht für Control-Taste, so bedeutet [CTRL+A], daß Sie die Control-Taste und die Taste »A« drücken müssen. Im folgenden steht:

[DOWN]	Taste neben rechtem Shift, Cursor unten	[SPACE]	Leertaste	[RVSON]	Control-Taste & 9
[UP]	Shift-Taste & Taste neben rechtem Shift; Cursor hoch	[SHIFT-Space]	Shift-Taste & Leertaste	[RVOFF]	Control-Taste & 0
[CLR]	Shift-Taste & 2. Taste ganz rechts oben	[F1] bis [F8]	Funktionstasten	[ORANGE]	Commodore-Taste & 1
[INST]	Shift-Taste & Taste ganz rechts oben	[RETURN]	Return-Taste	[BROWN]	Commodore-Taste & 2
[HOME]	2. Taste von ganz rechts oben	[BLACK]	Control-Taste & 1	[LIG.RED]	Commodore-Taste & 3
[DEL]	Taste ganz rechts oben	[WHITE]	Control-Taste & 2	[GREY 1]	Commodore-Taste & 4
[RIGHT]	Taste ganz rechts unten	[RED]	Control-Taste & 3	[GREY 2]	Commodore-Taste & 5
[LEFT]	Shift-Taste & Taste unten rechts	[CYAN]	Control-Taste & 4	[LIG.GREEN]	Commodore-Taste & 6
		[PURPLE]	Control-Taste & 5	[LIG.BLUE]	Commodore-Taste & 7
		[GREEN]	Control-Taste & 6	[GREY 3]	Commodore-Taste & 8
		[BLUE]	Control-Taste & 7		
		[YELLOW]	Control-Taste & 8		

Tabelle 1. Die Steuerbefehle in den Listings

Diese Zahlen dürfen Sie NICHT mit abtippen.
Als Beispiel sehen Sie Bild 1. Am rechten Rand jeder Spalte sehen Sie die Prüfsummen in eckigen Klammern.
Damit sind wir beim zweiten wichtigen Punkt: Sehen Sie sich die Zeile 240 von Listing 2 genauer an. Nach dem ersten Anführungszeichen nach dem PRINT-Befehl sehen Sie eine geschweifte Klammer { }. Immer, wenn Sie in einem unserer Listings diese Klammern sehen, dürfen Sie das, was innerhalb der Klammern steht, nicht eintippen. Sie müssen die entsprechende Taste drücken. Beispiel:
10 PRINT "{CLR}"
bedeutet: Nach dem Anführungszeichen die »Bildschirm-löschen«-Taste drücken (<SHIFT CLR/HOME>). In Tabelle 1 sehen Sie eine Zusammenfassung aller möglichen Steuertasten mit dem entsprechenden Klartext.
Weiterhin sehen Sie in Listing 2 (MSE) in Zeile 341 ein unterstrichenes »O« nach dem »P«. Das bedeutet, daß Sie ein »O« zusammen mit der SHIFT-Taste drücken müssen, also <SHIFT O>. Wenn ein Zeichen »überstrichen« ist, müssen Sie dieses zusammen mit der CBM-Taste eingeben. Die CBM-Taste befindet sich ganz links unten auf der Tastatur und hat die Aufschrift »C=«.

```
100 REM DIESES PROGRAMM ERZEUGT DEN          <210>
110 REM MSE V1.1 AUF DISKETTE.               <039>
120 REM BESITZER EINER DATASETTE             <178>
130 REM MUESSEN DIE '8' AM ENDE VON          <145>
140 REM ZEILE 343 IN EINE '1' AENDERN!       <176>
150 REM                                      <212>
230 IF PEEK(44)<>32 THEN PRINT"{CLR}SIE HA
    BEN VERGESSEN, DIE POKES EINZUGE- BEN!
    ":END                                    <050>
240 PRINT"{CLR}";:DIM H(75):FOR I=0 TO 9     <042>
250 H(48+I)=I:H(65+I)=I+10:NEXT:Z=1000       <136>
260 FOR I=2048 TO 3755 STEP 20:PRINT"{HOME
    }ICH LESE ZEILE:"Z                       <253>
261 FOR N=0 TO 19:READ A$:IF LEN(A$)<>2 TH
    EN 900                                   <062>
262 IF PEEK(63)+PEEK(64)*256<>Z THEN 800     <011>
270 H=ASC(LEFT$(A$,1)):L=ASC(RIGHT$(A$,1))   <199>
280 D=H(H)*16+H(L):S=S+D:POKE I+N,D          <165>
290 NEXT:READ V:IF S<>V THEN 900             <139>
300 S=0:Z=Z+1:NEXT:R=PEEK(2111):H=PEEK(210
    6)                                       <126>
301 POKE 53280,R:POKE 53281,H:POKE 646,R:P
    RINT"{CLR}DIE DATA-ZEILEN SIND FEHLERF
    REI!"                                    <080>
302 PRINT"SIE KOENNEN NUN DIE FARBEN DES M
    SE"                                      <209>
303 PRINT"EINSTELLEN.":PRINT"{2DOWN,SPACE,
    RVSON}DRUECKEN SIE <1>, <2> ODER <9>     <205>
304 PRINT"{DOWN,2SPACE}<1> - RAHMEN-/SCHRI
    FTFARBE                                  <013>
305 PRINT"{2SPACE}<2> - HINTERGRUNDFARBE     <233>
306 PRINT"{DOWN,2SPACE}<9> - FARBEN UEBERN
    EHMEN                                    <158>
307 PRINT"{2DOWN}FARBE <1>:"R:PRINT"FARBE
    <2>:"H                                   <066>
308 GET A:IF A=0 THEN 308                    <210>
309 IF A=1 THEN R=(R+1)AND 15                <098>
310 IF A=2 THEN H=(H+1)AND 15                <086>
311 IF A=9 THEN 340                          <217>
312 GOTO 301                                 <034>
340 POKE 2106,H:POKE 2111,R                  <153>
342 POKE 631,19:POKE 632,13:POKE 198,2       <135>
343 PRINT"{CLR}SAVE"CHR$(34)"MSE V1.1"CHR$
    (34)",8                                  <091>
344 POKE 43,1:POKE 44,8:POKE 45,172:POKE 4
    6,14:END                                 <140>
800 PRINT"{CLR,RVSON}SIE HABEN ZEILE"Z"{LE
    FT,SPACE}VERGESSEN:":A=PEEK(646)AND 15   <124>
810 POKE 646,PEEK(53281)AND 15:PRINT"LIST"
    Z-2"-"Z+2:POKE 646,A                     <224>
820 GOTO 920                                 <082>
900 PRINT"{CLR,RVSON}SIE HABEN EINEN TIPPF
    EHLER GEMACHT:":A=PEEK(646)AND 15        <154>
910 POKE 646,PEEK(53281)AND 15:PRINT"LIST"
    Z:POKE 646,A                             <173>
920 POKE 631,19:POKE 632,17:POKE 633,13:PO
    KE 198,3:END                             <126>
1000 DATA 00,0B,08,0A,00,9E,32,30,36,31,00
     ,00,00,A2,08,A9,36,85,A4,A9, 1247       <119>
1001 DATA 08,85,A5,A9,00,85,A6,A9,B0,85,A7
     ,A0,00,B1,A4,91,A6,C8,D0,F9, 2888       <054>
1002 DATA E6,A5,E6,A7,CA,D0,F2,A9,36,85,01
     ,4C,00,B0,20,D1,B1,A9,00,8D, 2781       <096>
1003 DATA 21,D0,A9,0F,8D,20,D0,8D,86,02,A0
     ,B3,A9,74,20,FF,B1,A0,B3,A9, 2679       <089>
1004 DATA B9,20,FF,B1,A0,00,20,CF,FF,99,01
     ,02,C8,C9,0D,D0,F5,88,F0,D2, 2912       <217>
1005 DATA C0,11,90,02,A0,10,8C,00,02,20,EA
     ,B1,A0,B3,A9,CF,20,FF,B1,20, 2327       <045>
1006 DATA 8E,B4,85,FC,85,62,20,8E,B4,85,FB
     ,85,61,20,A7,B4,D0,20,A0,B3, 2864       <199>
1007 DATA A9,E5,20,FF,B1,20,8E,B4,85,60,20
     ,8E,B4,85,5F,20,A7,B4,D0,0A, 2624       <091>
1008 DATA A5,61,C5,5F,A5,62,E5,60,90,06,20
     ,43,B3,4C,3A,B0,A9,AA,A0,00, 2379       <167>
1009 DATA EA,EA,E6,FB,D0,02,E6,FC,20,3F,B2
     ,90,EF,4C,FB,B4,A2,02,86,58, 3190       <041>
1010 DATA A9,A6,A0,9D,20,F2,B1,20,E4,FF,F0
     ,FB,C9,30,90,0C,C9,47,B0,08, 2970       <231>
1011 DATA C9,3A,90,0B,C9,41,B0,07,C9,14,D0
     ,0F,4C,0B,B1,20,D2,FF,A6,58, 2322       <121>
1012 DATA 95,F7,C6,58,D0,D2,60,AE,8D,02,F0
     ,26,C9,0C,D0,03,4C,0B,B6,C9, 2685       <057>
1013 DATA 13,D0,03,4C,8B,B5,C9,0D,D0,03,4C
     ,BA,B4,C9,10,D0,03,4C,68,B5, 2282       <225>
1014 DATA C9,0E,D0,06,20,5F,B4,4C,64,B1,4C
     ,92,B0,A5,F9,20,02,B1,0A,0A, 2132       <208>
1015 DATA 0A,0A,85,F9,A5,F8,20,02,B1,05,F9
     ,60,C9,3A,90,02,69,08,29,0F, 1950       <092>
1016 DATA 60,A6,59,E0,08,90,1F,A6,58,E0,02
     ,B0,06,20,D2,FF,4C,8E,B0,C6, 2509       <188>
1017 DATA 59,A0,14,A9,92,20,F2,B1,CA,D0,FA
     ,84,57,68,68,4C,8B,B1,A6,D3, 2891       <197>
1018 DATA E0,08,B0,03,4C,92,B0,20,D2,FF,A6
     ,58,E0,02,90,09,C6,59,20,D2, 2468       <049>
1019 DATA FF,C6,58,D0,F9,4C,8E,B0,48,4A,4A
     ,4A,4A,20,59,B1,68,29,0F,C9, 2419       <035>
1020 DATA 0A,90,02,69,06,69,30,4C,D2,FF,A2
     ,FC,9A,20,D1,B1,20,48,B2,20, 2261       <073>
1021 DATA EA,B1,20,9F,B2,A5,FC,20,4E,B1,A5
     ,FB,20,4E,B1,20,ED,B1,A9,3A, 2860       <148>
1022 DATA A0,20,20,F2,B1,A9,00,85,59,20,8E
     ,B0,20,ED,B1,A4,59,20,EF,B0, 2530       <233>
1023 DATA 91,FB,C8,84,59,C0,08,90,EC,20,10
     ,B2,A9,12,20,D2,FF,20,8E,B0, 2657       <105>
1024 DATA 20,EF,B0,C5,FF,F0,0D,20,43,B3,A9
     ,14,A0,14,20,F2,B1,4C,A2,B1, 2665       <034>
1025 DATA A9,92,20,D2,FF,20,33,B2,20,E0,B2
     ,20,3F,B2,90,9F,4C,8B,B5,A9, 2648       <123>
1026 DATA 93,20,D2,FF,A2,00,A9,03,9D,00,D8
     ,9D,00,D9,9D,00,DA,9D,00,DB, 2476       <237>
1027 DATA E8,D0,EF,60,A9,0D,2C,A9,20,4C,D2
     ,FF,20,D2,FF,98,4C,D2,FF,20, 2965       <160>
1028 DATA E4,FF,F0,FB,60,84,5D,85,5C,A0,00
     ,B1,5C,F0,06,20,D2,FF,C8,D0, 3100       <077>
1029 DATA F6,60,A5,FB,85,5A,A0,00,84,5B,B1
     ,FB,18,65,5A,85,5A,90,02,E6, 2606       <156>
1030 DATA 5B,06,5A,26,5B,C8,C0,08,90,EC,A5
     ,5A,65,5B,85,FF,60,18,A5,FB, 2467       <219>
1031 DATA 69,08,85,FB,90,02,E6,FC,60,A5,FB
     ,C5,5F,A5,FC,E5,60,60,A0,B3, 3106       <183>
1032 DATA A9,FB,20,FF,B1,A0,01,B9,00,02,20
     ,D2,FF,CC,00,02,C8,90,F4,A9, 2692       <098>
1033 DATA 14,ED,00,02,AA,20,ED,B1,CA,D0,FA
     ,A5,62,20,4E,B1,A5,61,20,4E, 2457       <060>
1034 DATA B1,20,ED,B1,A5,60,20,4E,B1,A5,5F
     ,20,4E,B1,EA,EA,EA,EA,EA,EA, 3122       <190>
1035 DATA EA,EA,24,5E,10,01,60,A9,12,20,D2
     ,FF,A2,28,20,ED,B1,CA,D0,FA, 2703       <087>
1036 DATA A9,92,4C,D2,FF,A5,D6,C9,16,B0,01
     ,60,A9,A0,85,A4,A9,78,85,A6, 2945       <204>
1037 DATA A9,04,85,A5,85,A7,A2,13,A0,27,B1
     ,A4,91,A6,88,10,F9,CA,F0,19, 2671       <208>
1038 DATA 18,A5,A4,69,28,85,A4,90,02,E6,A5
     ,18,A5,A6,69,28,85,A6,90,E0, 2503       <251>
1039 DATA E6,A7,4C,B6,B2,A9,91,4C,D2,FF,A9
     ,0F,8D,18,D4,A9,00,8D,05,D4, 2776       <000>
1040 DATA A9,F7,8D,06,D4,A9,11,8D,04,D4,A9
     ,32,8D,01,D4,A9,00,8D,00,D4, 2413       <126>
1041 DATA A0,80,20,09,B3,A9,10,8D,04,D4,60
     ,A2,FF,CA,D0,FD,88,D0,F8,60, 2914       <240>
1042 DATA A9,0F,8D,18,D4,A9,2D,8D,05,D4,A9
     ,A5,8D,06,D4,A9,21,8D,04,D4, 2385       <119>
1043 DATA A9,07,8D,01,D4,A9,05,8D,00,D4,A0
```

```
        ,FF,2Ø,Ø9,B3,A9,2Ø,8D,Ø4,D4, 225Ø    <Ø78>
1Ø44 DATA A9,ØØ,8D,Ø1,D4,8D,ØØ,D4,6Ø,38,2Ø
        ,FØ,FF,8A,48,98,48,18,AØ,Ø6, 2179    <175>
1Ø45 DATA A2,18,2Ø,FØ,FF,AØ,B4,A9,ØA,2Ø,FF
        ,B1,2Ø,12,B3,2Ø,E4,FF,FØ,FB, 2931    <Ø93>
1Ø46 DATA A2,1D,A9,14,2Ø,D2,FF,CA,DØ,FA,68
        ,A8,68,AA,18,4C,FØ,FF,ØD,ØD, 27Ø4    <Ø88>
1Ø47 DATA ØD,2Ø,2Ø,2Ø,2Ø,2Ø,2Ø,2Ø,4D,41,53
        ,43,48,49,4E,45,4E,53,5Ø,52, 1144    <216>
1Ø48 DATA 41,43,48,45,2Ø,2D,2Ø,45,44,49,54
        ,4F,52,2Ø,ØD,ØD,2Ø,2Ø,2Ø,2Ø, 1Ø23    <Ø38>
1Ø49 DATA 2Ø,2Ø,2Ø,2Ø,56,4F,4E,2Ø,4E,2E,4D
        ,41,4E,4E,2Ø,26,2Ø,44,2E,57, 1128    <2Ø6>
1Ø5Ø DATA 45,49,4E,45,43,4B,ØØ,ØD,ØD,ØD,2Ø
        ,2Ø,2Ø,5Ø,52,4F,47,52,41,4D, 11Ø2    <117>
1Ø51 DATA 4D,4E,41,4D,45,2Ø,3A,2Ø,ØØ,ØD,ØD
        ,2Ø,2Ø,2Ø,53,54,41,52,54,41, 1Ø73    <Ø95>
1Ø52 DATA 44,52,45,53,53,45,2Ø,3A,2Ø,24,ØØ
        ,ØD,ØD,2Ø,2Ø,2Ø,45,4E,44,41, 1Ø14    <129>
1Ø53 DATA 44,52,45,53,53,45,2Ø,2Ø,2Ø,3A,2Ø
        ,24,ØØ,92,Ø1,Ø1,5Ø,52,4F,47, 1136    <228>
1Ø54 DATA 52,41,4D,4D,2Ø,3A,2Ø,ØØ,12,2Ø,2Ø
        ,2A,2A,2A,2Ø,46,41,4C,53,43, 1Ø24    <Ø27>
1Ø55 DATA 48,45,2Ø,45,49,4E,47,41,42,45,2Ø
        ,2A,2A,2A,2Ø,2Ø,92,ØØ,ØD,ØD, 1Ø58    <Ø98>
1Ø56 DATA 2A,2A,2A,2Ø,45,4E,44,45,2Ø,2A,2A
        ,2A,ØØ,13,Ø1,2Ø,2Ø,12,44,92, 916     <153>
1Ø57 DATA 49,53,4B,2Ø,4F,44,45,52,2Ø,12,54
        ,92,41,5Ø,45,ØD,ØØ,13,2Ø,2Ø, 1151    <Ø35>
1Ø58 DATA 49,2F,4F,2Ø,2D,2Ø,46,45,48,4C,45
        ,52,ØØ,2Ø,D1,B1,2Ø,48,B2,AØ, 16Ø6    <Ø12>
1Ø59 DATA B3,A9,CF,2Ø,FF,B1,2Ø,8E,B4,85,FC
        ,2Ø,8E,B4,85,FB,C5,61,A5,FC, 32Ø7    <251>
1Ø6Ø DATA E5,62,9Ø,23,A5,FB,C5,5F,A5,FC,E5
        ,6Ø,BØ,19,2Ø,A7,D4,DØ,14,6Ø, 286Ø    <112>
1Ø61 DATA 2Ø,A7,B4,FØ,ØC,85,F9,2Ø,A7,B4,FØ
        ,Ø5,85,F8,4C,EF,BØ,68,68,2Ø, 2749    <Ø88>
1Ø62 DATA 43,B3,4C,5F,B4,2Ø,CF,FF,C9,4C,DØ
        ,Ø9,2Ø,D1,B1,2Ø,48,B2,4C,ØB, 2372    <Ø46>
1Ø63 DATA B6,C9,ØD,6Ø,A9,ØØ,85,5E,2Ø,5F,B4
        ,2Ø,EA,B1,2Ø,ØD,B5,24,5E,3Ø, 2Ø42    <12Ø>
1Ø64 DATA Ø5,2Ø,E4,FF,FØ,FB,2Ø,E1,FF,FØ,26
        ,2Ø,9F,B2,24,5E,1Ø,Ø9,2Ø,4E, 2435    <198>
1Ø65 DATA B5,2Ø,ØD,B5,2Ø,6Ø,B5,2Ø,33,B2,2Ø
        ,3F,B2,9Ø,D7,AØ,B4,A9,28,2Ø, 219Ø    <2Ø7>
1Ø66 DATA FF,B1,2Ø,E4,FF,C9,ØD,DØ,F9,A9,ØØ
        ,85,5E,A5,61,85,FB,A5,62,85, 3Ø56    <24Ø>
1Ø67 DATA FC,2Ø,EØ,B2,4C,64,B1,A5,FC,2Ø,4E
        ,B1,A5,FB,85,FF,2Ø,4E,B1,A9, 3ØØ3    <221>
1Ø68 DATA 2Ø,AØ,3A,2Ø,F2,B1,AØ,ØØ,2Ø,ED,B1
        ,B1,FB,2Ø,4E,B1,C8,CØ,Ø8,9Ø, 2566    <Ø7Ø>
1Ø69 DATA F3,2Ø,ED,B1,24,5E,3Ø,Ø3,A9,12,2C
        ,A9,2Ø,2Ø,D2,FF,2Ø,1Ø,B2,A5, 219Ø    <Ø59>
1Ø7Ø DATA FF,2Ø,4E,B1,A9,92,2Ø,D2,FF,4C,EA
        ,B1,A9,FF,85,B8,85,B9,A9,Ø4, 3Ø73    <Ø29>
1Ø71 DATA 85,BA,2Ø,CØ,FF,A2,FF,4C,C9,FF,2Ø
        ,CC,FF,A9,FF,4C,C3,FF,2Ø,5F, 3315    <189>
1Ø72 DATA B4,A9,8Ø,85,5E,2Ø,4E,B5,2Ø,48,B2
        ,A2,24,A9,2D,2Ø,D2,FF,CA,DØ, 2596    <111>
1Ø73 DATA FA,2Ø,EA,B1,2Ø,EA,B1,2Ø,6Ø,B5,4C
        ,C1,B4,2Ø,B8,B5,A6,5F,A4,6Ø, 2812    <Ø15>
1Ø74 DATA A9,61,2Ø,D8,FF,BØ,ØA,2Ø,B7,FF,29
        ,BF,DØ,Ø3,4C,FB,B4,A9,Ø1,2Ø, 2577    <2Ø1>
1Ø75 DATA C3,FF,2Ø,68,B6,AØ,B4,A9,4F,2Ø,FF
        ,B1,2Ø,F9,B1,4C,FB,B4,2Ø,68, 2921    <237>
1Ø76 DATA B6,A9,37,AØ,B4,2Ø,FF,B1,2Ø,F9,B1
        ,A2,Ø8,C9,44,FØ,Ø6,A2,Ø1,C9, 2717    <213>
1Ø77 DATA 54,DØ,F1,A9,Ø1,A8,2Ø,BA,FF,AØ,ØØ
        ,EØ,Ø1,FØ,1A,A9,4Ø,8D,2Ø,Ø2, 24Ø3    <1Ø1>
1Ø78 DATA A9,3A,8D,21,Ø2,B9,Ø1,Ø2,99,22,Ø2
        ,C8,CC,ØØ,Ø2,9Ø,F4,C8,C8,DØ, 2182    <127>
1Ø79 DATA ØC,B9,Ø1,Ø2,99,2Ø,Ø2,C8,CC,ØØ,Ø2
        ,DØ,F4,98,A2,2Ø,AØ,Ø2,4C,BD, 2Ø18    <Ø25>
1Ø8Ø DATA FF,2Ø,B8,B5,A5,BA,C9,Ø8,9Ø,33,A6
        ,B9,86,57,A9,Ø1,2Ø,C3,FF,A9, 28ØØ    <Ø22>
1Ø81 DATA 6Ø,85,B9,2Ø,CØ,FF,BØ,28,A5,BA,2Ø
        ,B4,FF,A5,B9,2Ø,96,FF,2Ø,A5, 2911    <Ø53>
1Ø82 DATA FF,85,61,A5,9Ø,4A,4A,BØ,13,2Ø,A5
        ,FF,85,62,2Ø,AB,FF,A5,57,85, 2663    <214>
1Ø83 DATA B9,A9,ØØ,2Ø,D5,FF,9Ø,Ø3,4C,A3,B5
        ,86,5F,84,6Ø,A5,BA,C9,Ø1,DØ, 2639    <131>
1Ø84 DATA ØA,AD,3D,Ø3,85,61,AD,3E,Ø3,85,62
        ,4C,FB,B4,A9,13,2Ø,D2,FF,A2, 23ØØ    <12Ø>
1Ø85 DATA 1C,2Ø,ED,B1,CA,DØ,FA,6Ø,ØØ,ØØ,ØØ
        ,ØØ,ØØ,ØØ,ØØ,ØØ,ØØ,ØØ,ØØ,ØØ, 123Ø    <143>
```


Listing 2. Der MSE-Lader

Der MSE

Der MSE dient zur Eingabe von Maschinensprache-Programmen. Als erstes müssen Sie den sogenannten »MSE-Lader« (Listing 2) abtippen. Dieser erzeugt erst das eigentliche MSE-Programm auf Diskette oder Kassette.

Wichtig: Vor dem Eintippen des MSE-Laders müssen Sie unbedingt ein paar Befehle eingeben (ohne Basic-Zeilennummer): POKE 44,32 : POKE 8192,0 : NEW

Jetzt können Sie beginnen, das Listing 2 abzutippen. Der MSE-Lader erkennt zwar, wenn Sie beim Eintippen der DATA-Zeilen einen Fehler gemacht haben, aber wenn Sie ganz sicher gehen möchten, sollten Sie den Checksummer vor dem Eintippen aktivieren. Die Prüfsummen für den MSE-Lader finden Sie am Ende der jeweiligen Programmzeilen.

Wenn Sie das Listing 2 nicht auf einmal abtippen möchten, müssen Sie vor jedem neuen Laden des Programms unbedingt die oben genannte POKE-Zeile eingeben!

Wenn Sie alles richtig gemacht haben und das Programm fehlerfrei abgetippt wurde, speichert es sich nach dem Starten selbst auf Diskette oder Kassette unter dem Namen »MSE V1.0«. Dieses fertige MSE-Programm laden Sie dann bei Bedarf wie ein normales Basic-Programm und starten es mit »RUN«.

So arbeitet man mit dem MSE

Als erstes möchte der MSE den Namen des zu bearbeitenden Programms wissen. Dieser steht in der ersten Zeile unserer MSE-Listings. Dann müssen Sie die Start- und Endadresse des Programms eingeben. Dies sind die letzten beiden, vierstelligen Hexadezimalzahlen in der ersten Zeile unserer Listings.

Wenn Sie ein Programm von Diskette oder Kassette laden wollen, um an einer bestimmten Stelle weiterzutippen oder noch eine Korrektur vorzunehmen, geben Sie auf die Frage nach der Startadresse ein »L« ein. Danach müssen Sie <D> oder <T> drücken, je nachdem, ob Sie von Diskette oder Kassette (»tape«) laden möchten. Wenn das Programm unter diesem Namen nicht auf der Diskette vorhanden ist oder ein sonstiger Ladefehler vorlag, meldet sich der MSE mit »I/O-ERROR«. In so einem Fall drücken Sie <RUN/STOP RESTORE> und geben einfach noch einmal »RUN« ein.

Beim Abtippen geben Sie nach und nach die abgedruckten Buchstaben und Zahlen des jeweiligen Listings ohne die Freiräume dazwischen ein. Wenn Sie in einer Zeile einen Tippfehler gemacht haben, meldet sich der MSE sofort mit einem Brummton und der Meldung »EINGABE-FEHLER«. Nach einem Druck auf die RETURN-Taste können Sie mit der DEL-Taste den Fehler korrigieren. Wenn Sie das gewünschte Programm vollständig eingegeben haben, speichert es der MSE automatisch auf Diskette oder Kassette.

Bei längeren Listings ist es unwahrscheinlich, daß Sie das komplette Programm auf einmal eingeben. Sie können Ihre bisherige Tipparbeit jederzeit durch <CTRL S> auf Diskette oder Kassette speichern und Ihr Werk später fortsetzen. Sie sollten sich dann allerdings im Heft markieren, wie weit Sie beim Abtippen gekommen sind! Später geben Sie dann nach dem Laden des ersten Programmteils <CTRL N> ein und auf die dann folgende Frage nach der Startadresse die Zeilennummer (Adresse), bei der Sie aufgehört haben zu tippen.

<CTRL M> erlaubt Ihnen jederzeit, Ihr Werk listen zu lassen. Durch <SPACE> können Sie weiterlisten lassen und durch <RUN/STOP> das Listen abbrechen.

Wenn Sie einen Drucker besitzen, können Sie das Programm auch mit <CTRL P> ausdrucken. Mit <CTRL L> wird das Programm noch einmal neu in Ihren C 64 geladen.

(F. Lonczewski/N. Mann/D. Weineck/tr)

Impressum

Herausgeber: Carl-Franz von Quadt, Otmar Weber

Geschäftsführender Chefredakteur: Michael Scharfenberger

Chefredakteur: Albert Absmeier
Leitender Redakteur: Georg Klinge
Redaktion: Markus Ohnesorg (og), Thomas Röder (tr), Gottfried Knechtel (kn), Peter Pfliegensdörfer (pd), Klaus Schrödl (sk), Norbert Jungmann (nj), Jörg Kähler (jk), Karsten Schramm (ks), René Beaupoil (rb), Peter Aurich (pa), Achim Hübner (ah), Roland Fieger (rf)

Layout: Leo Eder (Leitung), Rolf Raß (Cheflayouter)
Andrea Miller, Katja Milles

Titelfoto: Jens Jancke

Titelgestaltung: Eva-Maria Sperlich

Produktionsleiter: Klaus Buck

Anzeigenverkaufsleitung: Ralph-Peter Rauchfuss

Anzeigenverkauf: Helmut Distl (398)

Auslandsrepräsentation:

Schweiz: Markt&Technik Vertriebs AG,
Kollerstr. 3, CH-6300 Zug,
Tel. 042-41 56 56, Telex: 862329

USA: M&T Publishing Inc.; 501 Galveston Drive Redwood City, CA 94063
Telefon: (415) 366-3600

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten werden, so muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt&Technik Verlag AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Mit der Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt&Technik Verlag AG verlegten Publikationen und dazu, daß Markt&Technik Verlag AG Geräte und Bauteile nach der Bauanleitung herstellen läßt und vertreibt oder durch Dritte vertreiben läßt. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Marketingleiter: Hans Hörl (114)

Vertriebsleiter: Helmut Grünfeldt (189)

Anzeigenverwaltung und Disposition: Lisa Landthaler (233)

Druck: SOV St. Otto-Verlag GmbH,
Laubanger 23, 8600 Bamberg

Bezugsmöglichkeiten: Leser-Service: Telefon (089) 4613-249. Bestellungen nimmt der Verlag oder jede Buchhandlung entgegen.

Preis: Das Einzelheft kostet DM 14,–

Vertrieb Handelsauflage: Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: Pegasus Buch- und Zeitschriften-Vertriebs GmbH, Hauptstätter Straße 96, 7000 Stuttgart 1, Telefon (0711) 6483-0

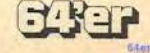
Verantwortlich:
Für redaktionellen Teil: Albert Absmeier
Für Anzeigen: Britta Fiebig

Redaktionsdirektor: Michael M. Pauly

Vorstand: Carl-Franz von Quadt, Otmar Weber

Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen:
Markt&Technik Verlag Aktiengesellschaft,
Hans-Pinsel-Straße 2, 8013 Haar bei München,
Telefon (089) 4613-0, Telex 5-22052

ISSN 0931-8933






U.S. GOLD